Expert Systems With Applications 112 (2018) 190–207

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Model checking of timed compatibility for mediation-aided web service composition: A three stage approach

Yanhua Du[a], Benyuan Yang[a], Hesuan Hu[b,*]

[a] School of Mechanical Engineering, University of Science and Technology Beijing, Beijing 100083, China
[b] School of Electro-Mechanical Engineering, Xidian University, Xi'an, Shaanxi 710071, China

ARTICLE INFO

Article history:
Received 21 July 2017
Revised 25 May 2018
Accepted 2 June 2018
Available online 15 June 2018

Keywords:
Mediation-aided service composition
Model checking
Timed compatibility
Temporal constraints
Petri nets

ABSTRACT

Currently, modeling and analyzing timed compatibility of Petri net based mediation-aided Web service composition by model checking are attracting increasing attention in the expert and intelligent systems community. However, existing methods cannot handle mediation-aided Web service composition involving complex mediation transitions, suffer from low efficiency owing to the larger size of time automata (*TA*) models, or are unable to automatically check temporal constraints whose activities do not include exchanged messages by observing *TA*. In this paper, we present a novel three-stage approach for analyzing timed compatibility of mediation-aided Web service composition via model checking. First, stage 1 treats each service in Petri net based mediation-aided Web service composition as a fragment. Second, stage 2 transforms fragments into a time automata net (*TAN*) based on structure transformation and interactive message transformation. Finally, stage 3 checks all types of temporal constraints. The main impact of our approach on expert and intelligent systems involves the following aspects: 1) mediation-aided service composition with complex mediation transitions can be dealt with; 2) compact *TAN* can be constructed for fragments in which the number of states and arcs is dramatically decreased and the verification time is extremely reduced compared with existing methods; and 3) temporal constraints whose activities have or do not have exchanged messages can be located in *TAN*. The main significance of our approach in the field of expert and intelligent systems is that it can greatly reduce the risk of making business decisions and the cost of handling temporal violations, and improves the innovation capability of enterprises.

1. Introduction

In recent years, Web service composition has become a hot topic in the expert and intelligent systems community (Bataineh, Bentahar, Menshawy, & Dssouli, 2017; Chang & Yuan, 2009; Jula, Sundararajan, & Othman, 2014; Kholy, Bentahar, Menshawy, Qu, & Dssouli, 2014; Rezaei, Chiew, Lee, & Aliee, 2014). Nowadays, the mediation-aided composition based on Petri nets (Du, Li, & Xiong, 2012; Du, Tan, & Zhou, 2015a; Li, Fan, Madnick, & Sheng, 2010; Tan, Fan, & Zhou, 2009; Tan, Fan, Zhou, & Tian, 2010; Xing, Li, Du, & Liang, 2016) is attracting increasing attention in Web service composition at the process level, because the formats of stored data and sequences of exchanged messages between Web services usually do not exactly match in practice (Guermouche et al., 2008; Du et al., 2012; Du et al., 2015a; Li et al., 2010; Mrissa, Sellami, Vettor, Benslimane, & Defude, 2013; Tan et al., 2009).

Usually, mediation-aided composition based on Petri nets is used to compose Web services via mediation transitions, i.e., forward mediation transition, merge mediation transition, and split mediation transition (Du et al., 2012; Tan et al., 2009) so that it can overcome incomplete and inaccurate message mapping between partially compatible Web services.

Currently, temporal constraints (Du, Xiong, Fan, & Li, 2011; Liu, Ni, Chen, & Yang, 2011a; Liu, Yang, Jiang, & Chen, 2011b; Liu, Yang, Yuan, & Chen, 2014) are regarded as an important aspect to ensure the correctness and quality of services (QoS) in Web services (abbreviated to services hereafter). They are usually set by implementation of activities and business rules, or are enforced by law. This leads to a new challenge in analyzing the timed compatibility of mediation-aided service composition. That is to say, to guarantee the correct execution of mediation-aided service compositions, we should check whether or not they are satisfied under the temporal constraints.

There are many techniques used to analyze temporal constraints in the research areas of workflow or service composition (Guermouche & Godart, 2010; Liu et al., 2011a; Liu et al., 2011b;

* Corresponding author.
E-mail addresses: duyanhua@ustb.edu.cn (Y. Du), yangbenyuan555@163.com (B. Yang), huhesuan@gmail.com (H. Hu).

https://doi.org/10.1016/j.eswa.2018.06.005
0957-4174/

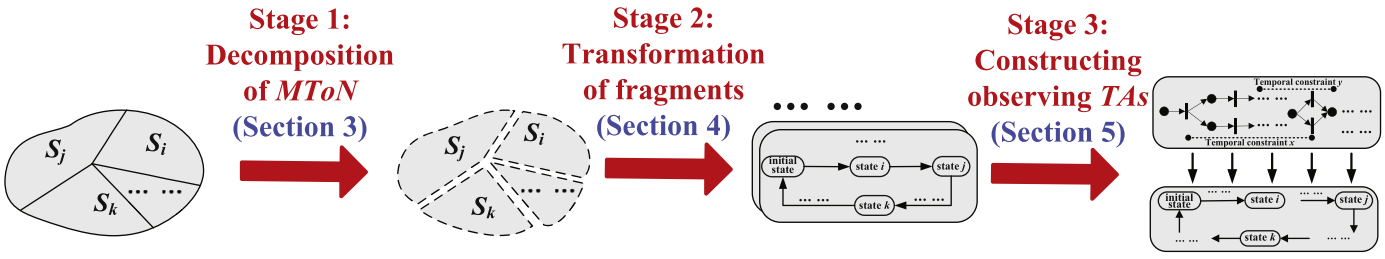


Fig. 1. Skeleton of our approach.

Santos, Li, Santos, & Korah, 2012, Bernardi & Campos, 2013; Du et al., 2015a; Du & Zhang, 2014; Gao & Singh, 2014; Liu et al., 2014; Ma, Zhang, & Lu, 2014, Xing et al., 2016). However, they cannot efficiently analyze temporal constraints for mediation-aided service composition because they do not fully investigate the message mismatches between services in a composition or suffer from low efficiency when checking the temporal constraints of mediation-aided service composition.

As a well-known verification technique, model checking (Bataineh et al., 2017; Bentahar, Yahyaoui, Kova, & Maamar, 2013; Du, Zhang, & Tan, 2013; Kokash, Krause, & deVink, 2010; Ponge, Boualem, Casati, & Toumani, 2010) is important to check temporal constraints or informal timed requirements of workflow or service composition. Model checking has been recognized by service communities for a long time and different methods have been developed. Thanks to these efforts, model checking has been one of the most successful verification approaches in the research area of service composition.

Based on model checking, there exist some research proposals which can be used to check temporal constraints for Petri net models of workflow or service composition, which can be divided into two aspects: *Transition – oriented method* (Du & Fan, 2010; Du et al., 2013; Gu & Kang, 2002) and *Partition – oriented method* (Du, Yang, & Tan, 2015b). The two methods can be used to analyze temporal constraints of mediation-aided service composition. However, they suffer from the following disadvantages: 1) they are unable to transform a Petri net based mediation-aided service composition involving complex mediation transitions to a *TAN*, because they cannot handle complex mediation transitions in the mediation-aided service composition; 2) *TANs* contain redundant states and arcs which will generate a large *TAN*, leading to low efficiency in analyzing temporal constraints by model checking; and 3) the temporal constraints whose activities do not involve exchanged messages cannot be located in a *TAN*, such that they cannot be automatically and efficiently checked by observing *TAs*.

In this paper, to overcome the disadvantages in *Transition – oriented method* and *Partition – oriented method*, we propose a new approach to analyzing timed compatibility of mediation-aided service composition based on Petri nets and model checking. The proposed approach is conducted in three stages, as depicted in Fig. 1. *Stage* 1 (Section 3): Each service in Petri net based mediation-aided service composition is treated as a whole and decomposed into one fragment. *Stage* 2 (Section 4): Fragments are transformed into a *TAN* with much fewer states and arcs, which can reduce the verification time of temporal constraints. *Stage* 3 (Section 5): Temporal constraints are located in *TANs* and observing *TAs* are constructed so that all temporal constraints can be automatically checked by a well-known model checking tool, i.e., UPPAAL (Ibrahim & Khalil, 2012; Mokadem, Berard, Gourcuff, Smet, & Roussel, 2010).

Compared with existing work, we believe our approach (called the *Service – oriented method* in this paper) is crucial and at a minimum will provide the following benefits:

1) Mediation-aided service composition with complex mediation transitions can be dealt with. This greatly expands the application scope of our approach, such that any service can be analyzed by enterprises in the current fiercely competitive business environment.

2) Compact *TAN* can be constructed for fragments in which the number of states and arcs is dramatically decreased and the verification time is extremely reduced compared with the existing methods. This greatly improves the management efficiency of enterprises, saving their decision-making time and cost.

3) Temporal constraints whose activities have or do not have exchanged messages can be located in *TAN*. This expands the verification ability to all temporal constraints, such that we can provide more powerful validation and analysis capability, and ensure that enterprises do not violate any law or regulation.

In the context of expert and intelligent systems, our approach is of significance to enterprise management in the current fiercely competitive business environment. It can greatly reduce the risk in business decisions and the cost of handling temporal violations, and improve the innovation capability of enterprises. In reality, especially in the community of expert and intelligent systems, our approach is crucial and it will bring at a minimum the following three benefits to enterprises: 1) it assists in the design and development of expert diagnosis or intelligent monitoring systems in enterprises. In particular, our approach is suitable for the complex case (service composition with complex mediation transitions), which is commonly viewed and frequently adapted in real-world business scenarios; 2) it expands the capabilities of existing legacy expert diagnosis or intelligent monitoring systems and improves their efficiency in analyzing temporal constraints. This is because our approach can check all types of temporal constraints, and their verification time is extremely reduced with the aid of a compact *TAN*; and 3) it can act as an expert guideline or manual to quickly and efficiently assist enterprises in conducting execution control and risk reduction, as the entire procedure can be applied in various model checking tools, e.g., UPPAAL.

The remainder of this paper is organized as follows. In Section 2, the problem statement along with a motivating scenario is presented. The decomposition of mediation-aided service composition is proposed in Section 3. In Section 4, we propose the transformation of fragments. Section 5 presents the procedure for checking temporal constraints by observing *TA*. In Section 6, the performance of our approach is analyzed. Section 7 discusses related work. Section 8 concludes this paper.

2. Motivating scenario and problem statement

2.1. Motivating scenario

Before presenting the formal problem, we focus on in this paper, we first explain the following symbols or definitions in Table 1, which will be used in the rest of this paper. Their formal definitions can be found in Tan et al. (2009), Du et al. (2015a), and Du et al. (2015b).

Table 1
Explanation of symbols or definitions.

Abbreviation	Definition	Descriptions
WF – Net	Workflow net	Formalizing services with set of places,transitions and arcs.
ToN	Timed open workflow Net	Extended from *WF – Net* by introducing interface to exchange messages and labeling minimum and maximum firing delays to each transition.
MM	Message mapping	A finite set of mapping rules denoted as $< source, target >$, where *source* and *target* are messages to send and receive.
MToN	Mediation-aided composition of *ToN*s	Modeling service composition with a set of *ToN*s and mediation transitions.
TA	Time automata	Finite automata with a finite set of real valued clocks. They are used to model and analyze timing behavior of services.
TAN	Time automata net	Set of *TA*s with shared variables and channels among *TA*s.

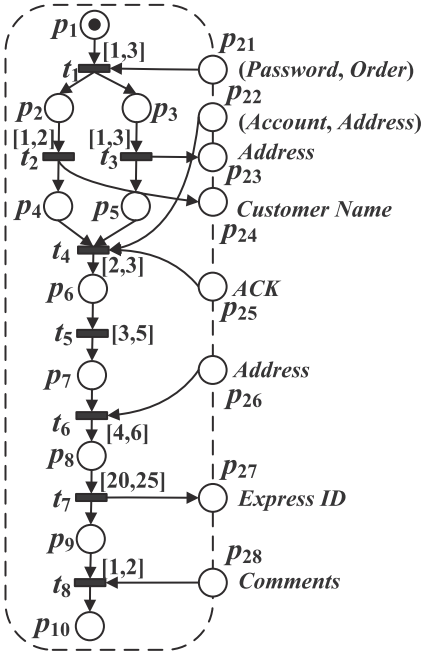


Fig. 2. *ToN* model of *OSS*.

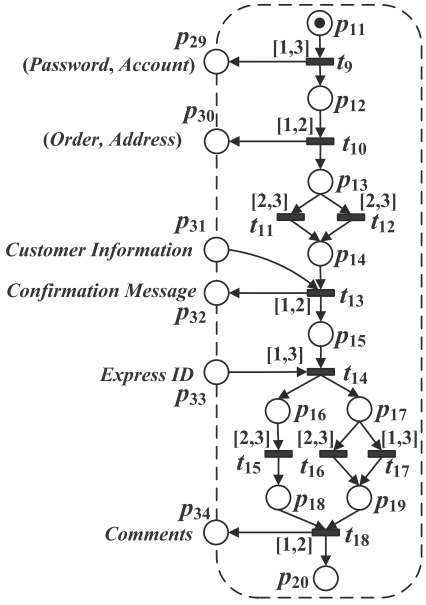


Fig. 3. *ToN* model of *CS*.

Table 2
Description of *OSS*.

Activity	Descriptions	Time intervals
t_1	Receive order	1–3 time units
t_2	Identify payment method	1–2 time units
t_3	Identify customer address	1–3 time units
t_4	Confirm and print order	2–3 time units
t_5	Pick and pack goods	3–5 time units
t_6	Take goods from warehouse	4–6 time units
t_7	Deliver goods	20–25 time units
t_8	Record comments	1–2 time units

Table 3
Description of *CS*.

Activity	Descriptions	Time intervals
t_9	Login account	1–3 time units
t_{10}	Submit order	1–2 time units
t_{11}	Choose credit as payment method	2–3 time units
t_{12}	Choose online banking as payment method	2–3 time units
t_{13}	Complete comments	1–2 time units
t_{14}	Receive goods	1–3 time units
t_{15}	Confirm goods	2–3 time units
t_{16}	Provide good comments	2–3 time units
t_{17}	Give the reason for negative feedback	1–3 time units
t_{18}	Submit comments	1–2 time units

Then, we present the motivating scenario which is excerpted and adapted from Du et al. (2015b). As depicted in Figs. 2 and 3, there are two services *OSS* (online shop service) and *CS* (customer service), which can be composed to coordinate executions and meet customer requirements. In this paper, we reuse the timed open workflow net (*ToN*) to formalize services at the *process level* based on Petri nets. Detailed information is provided in Tables 2 and 3.

Mediation-aided composition of *ToN*s (*MToN*) indicates composing *ToN*s by mediation transitions (i.e., a mediation net (*MN*)) based on message mapping of *ToN*s so that it can overcome the incomplete and inaccurate message mapping between partially compatible services. For example, the message (*Password, Account*) of p_{29} and (*Order, Address*) of p_{30} in *CS* do not directly match completely and accurately with (*Password, Order*) of p_{21} and (*Account, Address*) of p_{22} of *OSS*. Therefore, mediation transitions are necessary to split and merge these messages.

Assume that the message mapping between service *OSS* and *CS* is depicted as follows.

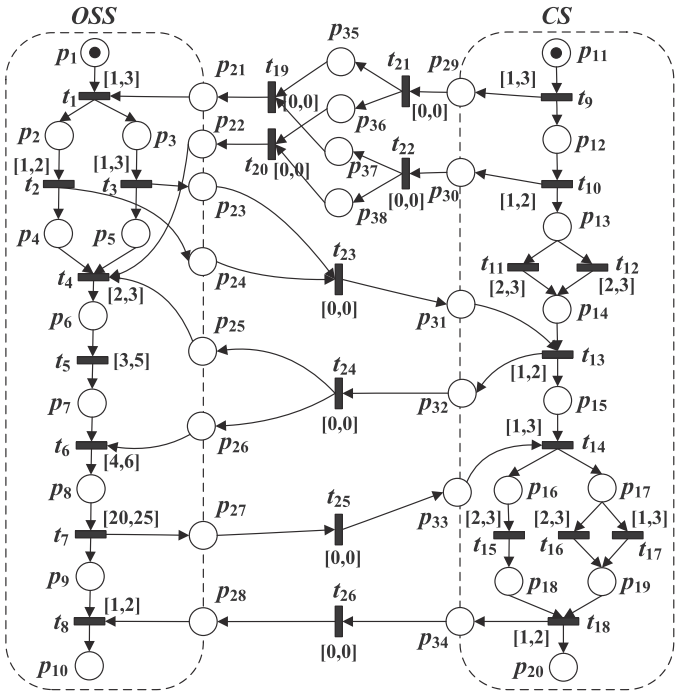


Fig. 4. Composition of *OSS* and *CS*.

Table 4
Description of temporal constraints.

NO.	Temporal constraints	Description
①	$D_R(t_4, t_7) \leq 45$	After confirming and printing the order, *OSS* must deliver goods to *CS* in no more than 45 time units.
②	$D_R(t_9, t_{13}) \leq 10$	The total time from logging in account to completing payment of *CS* must not exceed 10 time units.
③	$D_R(t_{14}, t_8) \leq 12$	After receiving goods from *OSS*, *CS* must complete the comments and send them to *OSS* within 12 time units.
④	$D_R(t_5, t_7) \leq 30$	After picking and packing goods, *OSS* must deliver goods in no more than 30 time units.

$< CS.(Password, Account).Password, OSS.(Password, Order).Password >$,
$<CS.(Password, Account).Account, OSS.(Account, Address).Account>$,
$< CS.(Order, Address).Order, OSS.(Password, Order).Order >$,
$< CS.(Order, Address).Address, OSS.(Account, Address).Address >$,
$< OSS.Address + OSS.Customer\ Name, CS.Customer\ Information >$,
$< CS.Confirmation\ Message, OSS.Ack + OSS.Address >$,
$< OSS.Express\ ID, CS.Express\ ID >$,
$< CS.Comments, OSS.Comments >$

Based on this we compose *OSS* and *CS* by *forward mediation transitions* (t_{25} and t_{26}), *merge mediation transitions* (t_{19}, t_{20}, and t_{23}), and *split mediation transitions* (t_{21}, t_{22}, and t_{24}) as shown in Fig. 4. For simplicity, we set time intervals of these mediation transitions as [0, 0].

To meet customer demands and gain a larger market share for online shop services, four temporal constraints are set in Table 4. To ensure the successful execution of mediation-aided composition of *OSS* and *CS*, we need to check whether the temporal constraints (i.e., timed compatibility) are satisfied or not. If all the temporal constraints in *MToN* are satisfied, then *MToN* is time-compatible. Otherwise, it is not time-compatible, i.e., violated, and some solutions should be implemented, such as (1) the change or replacement of service partners; (2) adjustment of service structure; (3) reduction of the execution time for services, and so on.

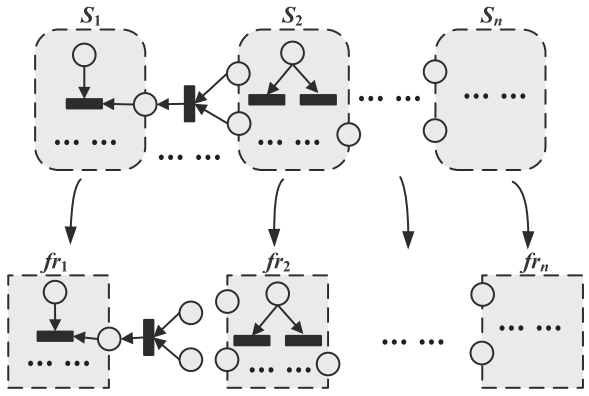


Fig. 5. Decomposing *MToN* to fragments based on services.

2.2. Problem statement

When the *Transition − oriented method* and *Partition − oriented method* are used to analyze the temporal constraints for the *MToN* in Section 2.1, they do not perform very well because of the following disadvantages: they cannot handle mediation-aided service composition involving complex mediation transitions, suffer from low efficiency owing to the larger size of *TA* models, or cannot automatically check temporal constraints whose activities do not involve exchanged messages by observing *TA*.

In this paper, the problem to be addressed is formally described as follows: Given (1) an *MToN* and (2) a set of temporal constraints denoted as *TCS*, where each one in *TCS* is defined as $D_R(t_i, t_j) \leq s$:

1) How can we decompose the *MToN* with complex mediation transitions to make them applicable in real-world business situations?

2) How to construct a more compact *TAN* so that the timed compatibility of *MToN* can be analyzed by model checking efficiently?

3) How to locate temporal constraints whose activities have or do not have input or output messages and then construct observing *TAs* for them, such that all the temporal constraints can be checked automatically and efficiently?

3. Decomposing *MTON*

As shown in Fig. 5, the essential idea of decomposing mediation-aided service composition is to treat each service in *MToN* as a fragment.

The reasons why we decompose the *MToN* into fragments by treating each service as a whole are as follows.

1) In a real-world business environment, services in the cloud platforms often need to be changed or replaced due to (Du et al., 2011; Santos et al., 2012) (1) a change, improvement, or replacement of services themselves; (2) real-time adjustment of service structure according to customer requirements; and (3) re-selection of services owing to business changes between services. Hence, it requires that services exhibit the capability to adjust quickly such that the property can be verified in timely fashion after any service is modified (Du et al., 2011; Liu et al., 2014; Santos et al., 2012). In this paper, we decompose the *MToN* into fragments from the perspective of services. When they are changed, we can directly locate the services which the modified activities belong to, and

then modify the services and their corresponding *TANs*. In this manner, the fragments can be modified and maintained more conveniently based on our approach, which can save labor and resources, and is of great importance to the optimization of the process in the design phase or for rapid deployment of business processes in run time. However, both *Transition – oriented method* and *Partition – oriented method* cannot deal with this issue.

2) There are always many mediation transitions, including complex ones among processes which can deal with partially compatible interfaces or interaction patterns (Li et al., 2010; Tan et al., 2009). However, the *Transition – oriented method* suffers from low efficiency as the number of services and activity of services increases. The *Partition – oriented method* is incapable of determining the specific boundaries of fragments (meaning that either input message or output message places will be contained in the generated service) such that it cannot decompose the *MToN* effectively while preserving the completeness of exchanged messages for each service process. To overcome this weakness, we decompose the *MToN* into fragments based on services. That is to say, first we obtain the input message places for each service. Then, we obtain all the mediation transitions between the output message places of one service and input message places of another, so that we can preserve the completeness of output messages for each service. Furthermore, the fragments will be more independent, such that it is possible for them to be modified without leading to other changes or cause the deadlock problem.

During the decomposition, we adopt the principle of treating the mediation transitions as input to the input places for services. Therefore, by extracting the input message place set of an *MToN*, we can obtain a unique decomposing model (fragments) for an *MToN*, i.e., all the mediation transitions which are inputs of some input message places will be explicitly contained in the generated fragments.

Definition 1. (*Input messages places*). P_I denotes the set of input message places, which are the input interfaces allowed to receive messages for a *MToN*. $\forall\ p\ \in P_I, {}^{\bullet}p = \varnothing$.

Definition 2. (*Output message places*). P_O is the set of output message places, which are the output interfaces allowed to send messages for a *MToN*. $\forall\ p\ \in\ P_O, p^{\bullet} = \emptyset$.

Definition 3. (*Fragment*). A fragment is a 6-tuple=($WF - Net$, D_I, P_I, P_O, *MT*, *FM*), where $WF - Net$ is a workflow net, D_I is the set of durations (execution times) of transitions, P_I is the set of input message places in Definition 1, P_O is the set of output message places in Definition 2, *MT* is a set of mediation transitions (i.e., forward, merge, and split mediation transitions) among P_O and P_I, and *FM* is a set of directed arcs linking *MT* with P_I.

The procedure for decomposing *MToN* into fragments is shown in Algorithm 1 We use $t^{\bullet}$ (${}^{\bullet}t$) to describe the postset (or preset) of t, which is the set of output (or input) places of t, and adopt $p^{\bullet}$ (${}^{\bullet}p$) to describe the postset (or preset) of p which is the set of output (or input) transitions of p, respectively (Du et al., 2011; Hu & Liu, 2015).

Algorithm 1 depicts our decomposition procedure by treating each service as a whole. The procedure of decomposing mediation-aided service composition into fragments is as follows: First, all of the mediation transitions in a *MToN* are extracted ($L4$). Then, the algorithm extracts all services and their input message place ($L5 \sim L6$). Third, the algorithm associates each mediation transition as well as the arcs connecting them to a service based on input message places ($L7 \sim L12$). Specifically, if a mediation transition mt_{ii} connects with an output message place of service s_j and

Algorithm 1: Decompose *MToN* into fragments.

```
1  Input: MToN;
2  Output: The set of fragment SF;
3  Initialization: MToN = { s_1, s_2, ... , s_n, MN }, set SF = ∅, and the
   mediation transition set MTS = ∅;
4  Extract all of the mediation transitions from MToN and put
   them in MTS={ mt_i1, mt_i2, ... , mt_in };
5  for i = 1; i ≤ n; i++ do
6      Obtain the input message place set OPM_i = { p_Minii, ... ,
       p_Minik } of s_i;
7      for j = 1; j ≤ m; j++ do
8          Take the input message place p_Minij from OPM_i and get
           mt_ii from MTS which satisfies mt_ii ∈ •p_Minij;
9          if mt_ii ∈ •p_Minij and mt_ii ∈ p_i• /* p_i is not an output
           message place of s_j, i ≠ j */ then
10             Traverse all the elements which are the post-set of
               p_i until we find a mediation transition mt_jj connect-
               ing with an output message place of service s_j, and
               then all of the elements between mt_ii and mt_jj are
               assigned to s_i;
11         else if mt_ii ∈ •p_Minij and mt_ii ∈ p_Moutij• then
12             Combine mt_ii and p_Moutij with s_i; /* p_Minij is an input
               message place of s_i and p_Moutij is an output message
               place of s_j, i ≠ j */
13     SF = SF ∪ { s_i }.
```

input message place of s_i, then it is assigned to s_i ($L11 \sim L12$). Otherwise, we need to traverse all the elements which are the post-set of mt_{ii} until we find a mediation transition mt_{jj} connecting with an output message place of service s_j, and assign each to s_i ($L9 \sim L10$). Finally, by repeating the above processes, we obtain all of the fragments ($L13$).

By assuming that there are n services and m input message places in the models, the time complexity of Algorithm 1 is $O(nm)$.

Take the *MToN* in Fig. 4 as an example. We first obtain the mediation transition set $MTS = \{ t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26} \}$. Second, we retrieve the *OSS* and its input message place set $OPM_{OSS} = \{ p_{21}, p_{22}, p_{25}, p_{26}, p_{28} \}$. Then, we take p_{21} from OPM_{OSS} and t_{19} from *MTS* which satisfy $t_{19} \in {}^{\bullet} t_{21}$. Next, we first select p_{35} and p_{37}, which satisfy $t_{19} \in p_{35}{}^{\bullet}$ and $t_{19} \in p_{37}{}^{\bullet}$, and then we find the mediation transitions t_{21} and t_{22} where $t_{21} \in {}^{\bullet} p_{35}$ and $t_{22} \in {}^{\bullet} p_{37}$, and combine t_{19}, t_{21}, t_{22}, p_{35} and p_{37} with *OSS* through the arcs connecting them. For $t_{21} \in p_{29}{}^{\bullet}$, $t_{22} \in p_{30}{}^{\bullet}$, because p_{29} and p_{30} are the output message places of *CS*, we combine them with *OSS*. Repeating the steps above, we combine t_{20}, t_{24}, t_{26}, p_{36}, p_{38}, p''_{29}, p''_{30}, p_{32}, and p_{34} with *OSS* through the arcs between them. Finally, we obtain the fragment of *OSS* as shown in Fig. 6(a). Similarly, we can obtain the fragment of *CS* as shown in Fig. 6(b).

If we adopt the *Partition – oriented method*, each service in a mediation-aided service composition is decomposed into several fragments. It is difficult to modify and maintain changed services in a real business environment, and mediation-aided service composition that includes complex mediation transitions cannot be decomposed. For example, we can see from Fig. 4 that the complex mediation transitions among places p_{21}, p_{22}, p_{29}, and p_{30} are combined by several basic mediation transitions, and cannot be decomposed by any traditional approaches.

According to Algorithm 1, any mediation-aided service composition can be decomposed into fragments by traversing and copying the mediation transitions. To confirm this, the following theorem is proved.

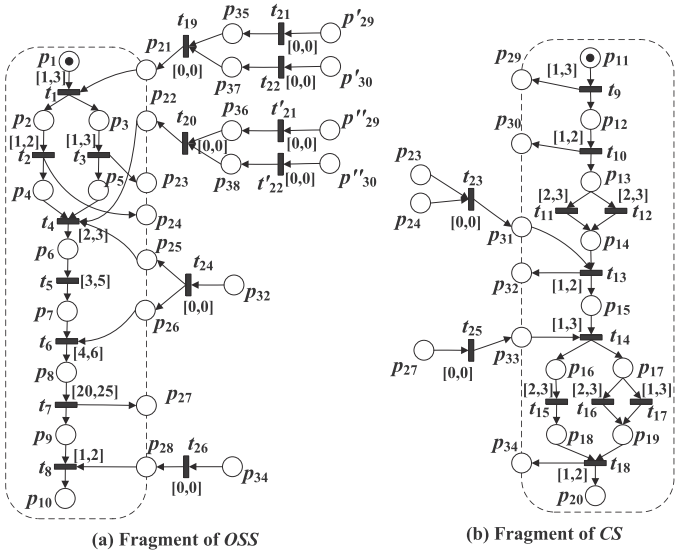


Fig. 6. Fragments of *MToN*.

Theorem 1. *Any mediation-aided service composition can be decomposed into fragments by traversing and copying the mediation transitions.*

Proof. Assume mt_i is an arbitrary mediation transition in mediation-aided service composition.

1) Suppose that mt_i is a basic mediation transition, such as a store/forward mediation transition, merge mediation transition, or split mediation transition. Based on the approach by Du et al. (2015b), we can traverse and copy those three basic mediation transitions;

2) Suppose that mt_i is a complex mediation transition which consists of basic ones. According to the approach in Du et al. (2015b), any basic mediation transition can be traversed and copied. Complex mediation transitions can also be traversed and copied because they are constructed by basic mediation transitions. If not, then the basic mediation transition is incorrect. This is contrary to the approach in Du et al. (2015b).

Therefore, any mediation-aided service composition can be decomposed into fragments by traversing and copying the mediation transitions. □

Note that the decomposition of mediation-aided service composition in this paper does not lead to the deadlock problem which usually exists in the area of flexible manufacturing systems (FMS) (Baruwa, Piera, & Guasch, 2015; Hu & Liu, 2015; Hu, Liu, & Yuan, 2016; Luo, Xing, Zhou, Li, & Wang, 2015). When we decompose mediation-aided service composition into fragments, the input or output messages of each service in the mediation-aided service composition are preserved by copying the corresponding input or output message places, and the exchanged messages of each service are never changed. Therefore, our decomposition approach does not cause the deadlock problem.

4. Transforming fragments

In this section, we first present the basic idea of transforming fragments to *TAN*. Then, we present the detailed procedure of automatic transformation.

4.1. Analysis of transforming fragments

Fragments are actually the sub-nets of Petri nets which include two types of elements: *structure* and *interactive message*.

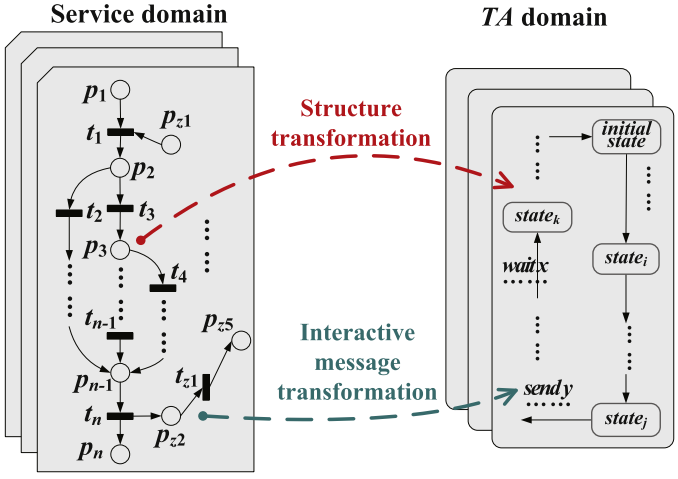


Fig. 7. Transforming fragments to *TAN*.

Table 5
Structure and interactive message transformation.

Transformation	Descriptions
Structure transformation	Transform starting places and paths in a fragment to states and arcs with clocks, invariant conditions, or guard conditions in a *TAN*.
Interactive message transformation	Transform input messages (or output messages) to channels and use *wait* (or *send*) channels to denote that a channel is waiting (or sending).

Therefore, we transform the fragments to *TAN* based on two types of transformations: structure and interactive message, as depicted in Fig. 7.

The structure describes the logic information of fragments, which includes starting places, paths, and time intervals of transitions. The interactive message includes the exchanged messages between fragments, which are labeled in the structures representing the exchanged messages between services.

Before we present the detailed procedure of automatic transformation, we first propose the concept of interactive message and path, which is used in the remainder of this paper.

Definition 4. (*Interactive message*). Interactive message describes the exchanged message mapping between fragments, in the form $< source, target >$. Specifically, the form $< p_s.m_a, p_t.m_b >$ is used to describe the message mapping, where p_s is *source* corresponding to a set of output message places and m_a is an output message in p_s; p_t is a *target* corresponding to a set of input message places, and m_b is an input message in p_t.

Definition 5. (*Path*). $\{ t_1, t_2, \ldots, t_k \}$ is said to be a path, if $\forall i$, $1 \leq i \leq k-1$, $\exists p \in P$, arcs (t_i, p) and (p, t_{i+1}) exist.

Then, we specify the structure and interactive messages transformation, as shown in Table 5.

1) *Structure transformation* is to transform the starting places of fragments to states and arcs with clocks in *TAN*, and paths of fragments to states with invariant conditions and arcs with guard conditions in *TAN*.

Specifically, the starting places of a fragment are transformed to initial states and the arcs with clocks, such as $x = 0$, $y = 0$ in *TAN* initially. For path, such as $path_x = \{ t_i, t_{i+1}, \ldots, t_k, t_{k+1}, \ldots, t_j \}$, assume that the time interval of each is $[sl_i, su_i]$, $[sl_{i+1}, su_{i+1}]$, ... , $[sl_k, su_k]$, $[sl_{k+1}, su_{k+1}]$, ... , and $[sl_j, su_j]$, respectively. (1) If all transitions in $path_x$ do not contain interactive messages, $path_x$ is

compressed and transformed to a state with invariant condition $x \le su_i + su_{i+1}$+... +$su_k + su_{k+1}$+... +su_j and an arc with guard conditions $x \ge sl_i + sl_{i+1}$+... +$sl_k + sl_{k+1}$+... +sl_j. (2) If there exists a transition t_k in $path_x$ which has interactive messages, then { t_i, t_{i+1}, ... , t_k } of $path_x$ is compressed and transformed to a state with invariant condition $x \le su_i + su_{i+1}$+... +su_k and an arc with guard conditions $x \ge sl_i + sl_{i+1}$+... +sl_k. Another part, { t_{k+1}, ... , t_j }, of $path_x$ is compressed and transformed to a state with invariant condition $y \le su_{k+1}$+... +su_j and an arc with guard conditions $y \ge sl_{k+1}$+... +sl_j. Because the paths in the fragments are compressed, the numbers of both states and arcs are dramatically decreased in the *TAN*, which is transformed from these fragments.

2) *Interactive message transformation* is to transform the message mapping between different services to channels which consist of a set of integer variables used to synchronize messages between *TA*s. This procedure includes two aspects: (1) each input message is transformed to a channel and then uses a *wait* channel to denote that a channel is waiting, and (2) each output message is transformed to a channel and then uses a *send* channel to denote that a channel is sending.

Specifically, for "*one to one*" pattern, where there is one input message and one output message, assuming the message mapping is in the form of $< p_s.m_a, p_{ti}.m_i >$, we use one *send* channel and one *wait* channel to realize the control logic and communication of messages in *TAN*; for "*one to more*" pattern, in which is one message split into multiple messages and the message mapping is in the form of $< p_s.m_a, p_{ti}.m_i$ + ... + $p_{tj}.m_j >$, we use one *send* channel and multiple *wait* channels; for "*more to one*" pattern to merge multiple messages into one message, and if the message mapping is in the form of $< p_{sm}.m_m$ + ... + $p_{sn}.m_n, p_t.m_d >$, we use multiple *send* channels and one *wait* channel. In addition, "*more to more*" pattern can be decomposed into "*one to more*" and "*more to one*" patterns as mentioned in Section 3, so we can deal with them by the above methods.

4.2. Automatic transforming procedure

The detailed procedure of structure transformation is presented in Algorithm 2 .

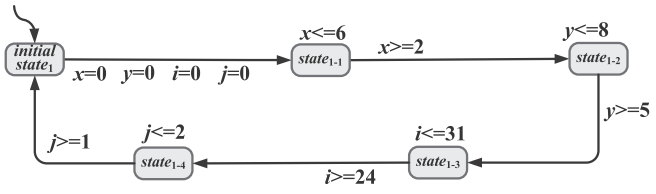
Algorithm 2 describes how to transform the structure of fragments. The procedure is as follows: starting places of fragments are transformed to the states and arcs with clocks initially (*L*6). Then, the algorithm obtains all of the paths in a fragment (*L*7) and transforms the paths of a fragment to the states with invariant conditions and arcs with guard conditions in *TAN* (*L*8 ~ *L*15). Specifically, for any path, if t_{ik} is the input of an input message place, the sub-net between starting transition t_{ij} and t_{ik} is transformed (*L*12 ~ *L*13); Otherwise, obtain the transition executed before t_{ik}, denoted as t_{ik-1} and then transform the sub-net between starting transitions t_{ij} and t_{ik-1} (*L*14 ~ *L*15). Finally, by repeating the above process, the algorithm returns a *TAN* without channels (*L*16).

By assuming that there are n fragments and m transitions in the models, the time complexity of Algorithm 2 is $O(nm^2)$.

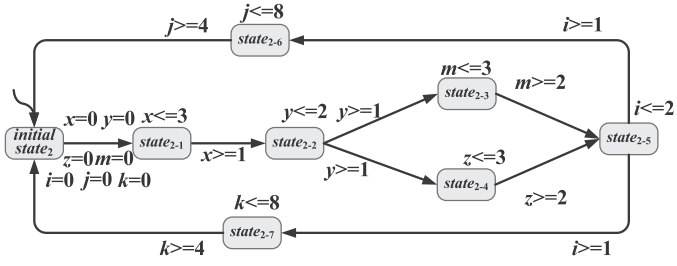
Consider the fragments of *MToN* in Fig. 6. We first take the fragment of *OSS* from Fig. 6(a). Second, the starting place p_1 is transformed to the *initial state*$_1$ and the arc from *initial state*$_1$ to $state_{1-1}$ with clocks $x = 0$, $y = 0$, $i = 0$, and $j = 0$ in Fig. 8(a). Third, we obtain the path { t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8 } in Fig. 6(a). Then, t_1, t_2, and t_3 are compressed and transformed to $state_{1-1}$ with invariant condition $x \le 6$ and the arc from $state_{1-1}$ to $state_{1-2}$ with guard condition $x \ge 2$ in Fig. 8(a). In the same way, we can transform t_4 and t_5 to $state_{1-2}$ with invariant condition $y \le 8$ and the arc from $state_{1-2}$ to $state_{1-3}$ with guard condition $y \ge 5$; t_6 and t_7 to $state_{1-3}$ with invariant condition $i \le 31$ and the arc from $state_{1-3}$ to $state_{1-4}$ with guard condition $i \ge 24$; and t_8 to $state_{1-4}$ with invariant condition $j \le 2$ and the arc from $state_{1-4}$ to *initial state*$_1$ with guard

Algorithm 2: Transform structure of fragments.

1 **Input**: The set of fragments *SF*;
2 **Output**: *TAN* without channels *TAN*′;
3 Initialization: *SF* = { fr_1, fr_2, ..., fr_n }, set *TAN*′ = ∅;
4 **for** $i = 1$; $i \le n$; i++ **do**
5 Denote the *TA* that will be transformed from fr_i as TA_i;
6 For starting place p_{ini} of fr_i, we construct a location *initial state* and the edge l_{z0} from it to location *state* z_0 with corresponding local clocks x=0, ..., y=0 in TA_i;
7 Obtain paths from starting place to ending place of fr_i, denoted as TS_i;
8 **for** $j = 1$; $j \le TS_i$.length; j++ **do**
9 Obtain the transition t_{ij};
10 **for** $k = 1$; $k \le TS_i$.length; k++ **do**
11 Obtain the transition except for t_{ij} which has input message places or output message places, denoted as t_{ik};
12 **if** ∃ $t_{ik} \in TS_i$ and $t_{ik} \in {}^\bullet p_{Mouti}$ **then**
13 The sub-net between transition t_{ij} and t_{ik} is transformed into a location *state* m_0 with invariant condition $x \le m$ and an arc l_{n0} from m_0 to location state n_0 with guard condition $x \ge n$ in TA_i, where m and n are the sum of the maximum and minimum time intervals of all the transitions between transitions t_{ij} and t_{ik}, respectively;
14 **else if** ∃ $t_{ik} \in TS_i$ and $t_{ik} \in p_{Mini}{}^\bullet$ **then**
15 Obtain the transition executed before t_{ik}, denoted as t_{ik-1}. Then we transform the sub-net between transitions t_{ij} and t_{ik-1} into a location state s_0 with invariant condition $y \le s$ and an edge l_{n0} from s_0 to location state t_0 with guard condition $y \ge t$ in TA_i, where s and t are the sum of maximum and minimum time intervals of all transitions between transitions t_{ij} and t_{ik-1}, respectively;
16 *TAN*′ = *TAN*′ ∪ { TA_i }.



(a) *TA* for fragment of *OSS* without channels



(b) *TA* for fragment of *CS* without channels

Fig. 8. *TAN*s for fragments of *OSS* and *CS* without channels.

condition $j \geq 1$, respectively. Finally, we can obtain the *TA* for the fragment of *OSS* without channels as shown in Fig. 8(a). Following the same process, we obtain the *TA* for the fragment of *CS* without channels, as shown in Fig. 8(b).

The structure transformation only obtains a *TAN* without channels which cannot be used to check properties. Therefore, we propose Algorithm 3 to realize the transformation of interactive messages.

Algorithm 3: Transform interactive messages of fragments.

1 **Input**: *MToN* and *TAN* without channels *TAN′*;
2 **Output**: *TAN*;
3 Initialization: $TAN = \emptyset$, $j = 1$, and denote the set of all message mappings as *MM* and set $MM = \emptyset$;
4 Obtain the set of mediation transitions $MTS = \{ mt_1, mt_2, ..., mt_n \}$ in *MToN*;
5 **for** $i = 1$; $i \leq n$; $i++$ **do**
6 Obtain the corresponding message mapping MM_i;
7 $MM = MM \cup \{ MM_i \}$;
8 **while** $MM \neq \emptyset$ **do**
9 Extract MM_j from *MM*;
10 **if** $MM_i = < p_s.m_a, p_t.m_b >$ **then**
11 Assume $t_s \in {}^\bullet p_s$, $t_t \in p_t {}^\bullet$, then we add *send* m_a to the corresponding arc of t_s and wait m_b to the corresponding arc of t_t in the *TAN*;
12 **else if** $MM_i = < p_s.m_a, p_{ti}.m_i + ... + p_{tj}.m_j >$ **then**
13 Assume $t_s \in {}^\bullet p_s$, $t_{ti} \in p_{ti} {}^\bullet$, ..., $t_{tj} \in p_{tj} {}^\bullet$, and then we add *send* m_a to the corresponding arc of t_s and *wait* m_i, ..., *wait* m_j to the corresponding arc of t_{ti}, ..., t_{tj} in the *TAN*;
14 **else if** $MM_i = < p_{sm}.m_m + ... + p_{sn}.m_n, p_t.m_d >$ **then**
15 Assume $t_{sm} \in {}^\bullet p_{sm}$, $t_{sn} \in {}^\bullet p_{sn}$, ..., $t_t \in p_t {}^\bullet$, and then we add *send* m_m, ..., *send* m_n to the corresponding arc of t_{sm}, ..., t_{sn} and *wait* m_d to the corresponding arc of t_t in the *TAN*;
16 **else if** MM_i is a complex message mapping **then**
17 Decompose it into basic messages and then set channels based on the above three patterns;
18 $MM = MM - MM_j$;
19 $j = j++$.

Algorithm 3 describes the procedure for transforming interactive messages of fragments. First, the algorithm obtains all of the mediation transitions and their message mapping (*L*4 ∼ *L*7). Then the algorithm transforms the messages to channels and use *send* or *wait* channels to denote that a channel is waiting or sending in the *TAN* to obtain a complete *TAN* (*L*8 ∼ *L*19). Specifically, *L*10 ∼ *L*11 transforms an interactive message by one to one pattern, *L*12 ∼ *L*13 transforms an interactive message by one to more pattern, *L*14 ∼ *L*15 transforms an interactive message by more to one pattern, and *L*16 ∼ *L*17 transforms an interactive message by more to more pattern.

By assuming that there are *n* mediation transitions in the models, the time complexity of Algorithm 3 is $O(n^2)$.

Reconsider the example of *TAN*s for fragments of *OSS* and *CS* without channels in Fig. 8. Based on Algorithm 3, we first obtain mediation transitions t_{19}, t_{20}, t_{21}, t_{22}, t'_{21}, t'_{22}, t_{23}, t_{24}, t_{25}, and t_{26}. Second, we obtain the message mapping $< p_{29}.m'_{29}, p_{21}.m_{21} >$, $< p_{29}.m''_{29}, p_{22}.m_{22} >$, $< p_{30}.m'_{30}, p_{21}.m_{21} >$, $< p_{30}.m''_{30}, p_{22}.m_{22} >$, $< p_{23}.m_{23} + p_{23}.m_{24}, p_{31}.m_{31} >$, $< p_{32}.m'_{32}, p_{25}.m_{25} + p_{26}.m_{26} >$, $< p_{27}.m_{27}, p_{33}.m_{33} >$, and $< p_{34}.m_{34}, p_{28}.m_{28} >$. Third, we take the first message mapping $< p_{29}.m'_{29}, p_{21}.m_{21} >$. Because $t_9 \in {}^\bullet p_{29}$, $t_1 \in p_{21} {}^\bullet$, we add the *send* m'_{29} in the arc from $state_{2\text{-}1}$ to $state_{2\text{-}2}$ in Fig. 9(b) and add *wait* m'_{29} in the arc from *initial* $state_1$ to $state_{1\text{-}1}$ in Fig. 9(a). In the same way, we add the *wait* m_{25} in the arc from *initial* $state_1$ to $state_{1\text{-}1}$, *send* m_{31}, *wait* m_{25}, *wait* m''_{29}, and *wait* m''_{30} in the arc from $state_{1\text{-}1}$ to $state_{1\text{-}2}$; *wait* m_{26} in the arc from $state_{1\text{-}2}$ to $state_{1\text{-}3}$, *send* m_{33}, and *wait* m_{28} in the arc from $state_{1\text{-}3}$ to $state_{1\text{-}4}$ to obtain the *TA* for the fragment of *OSS*, as depicted in Fig. 9(a). In the same way, we add the corresponding channels and obtain the *TA* for the fragment of *CS* as depicted in Fig. 9(b). To compare our method with the *Transition – oriented method* and *Partition – oriented method*, we also apply these two methods to the fragments of *MToN* in Fig. 6. Table 6 shows the *TA*s, states, and arcs generated by *Transition – oriented method*, *Partition – oriented method*, and our *Service – oriented method*.

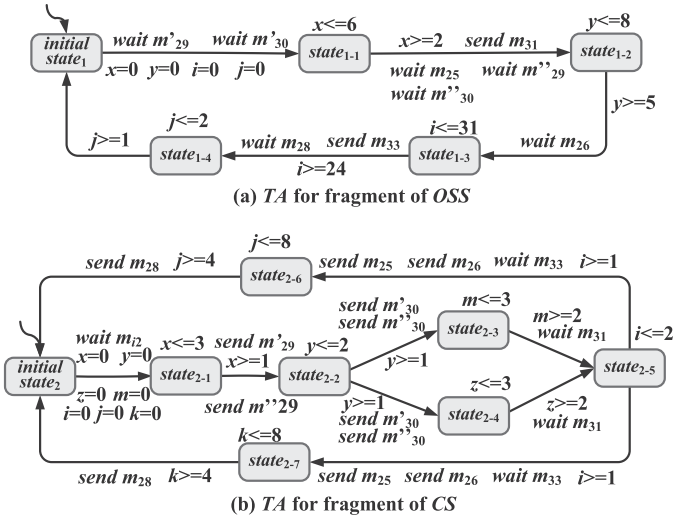


Fig. 9. *TAN* s for fragments of *OSS* and *CS*.

Table 6
Comparison of three methods.

Methods	TAs	States	Arcs
Transition – oriented method	26	52	52
Partition – oriented method	6	20	22
Service – oriented method	2	13	15

As shown in Table 6, there are two *TA*s, 13 states, and 15 arcs in the *TAN* for *MToN* as depicted in Fig. 9. However, based on the *Transition – oriented method*, we need to construct 26 *TA*s with 52 states and 52 arcs, as each transition is transformed into a *TA* with two states and two arcs connecting them. If we use the *Partition – oriented method*, the *TAN* transformed from fragments contains redundant states and arcs which may generate a larger *TAN*, leading to low efficiency. For example, because t_4 and t_5 in Fig. 4 are the ending places of the fragment between p_1 and $\{ p_4, p_5 \}$, and are also the starting places of another fragment between $\{ p_4, p_5 \}$ and p_7, we need to repeatedly transform them if we use *Partition – oriented method* which will lead to 6 *TA*s, 20 states, and 22 arcs in the *TAN*.

Therefore, the quantities of states/arcs in the *TAN* obtained by our approach are less than those obtained by the *Transition – oriented method* and *Partition – oriented method*.

5. Checking by observing *TA*s

In this section, we first propose how to locate temporal constraints from a *TAN*. Then we present the procedure for checking temporal constraints by observing *TA*s.

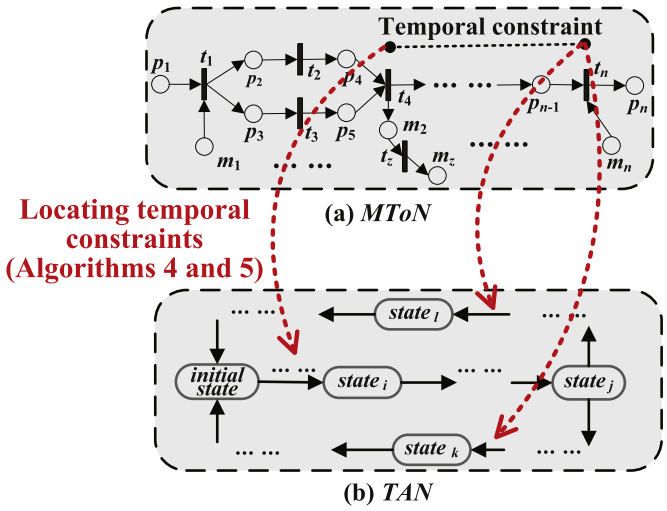


Fig. 10. Locating temporal constraints.

5.1. Locating temporal constraints from MToN to TAN

After decomposing the *MToN* into fragments and transforming them to the *TAN*, observing *TA*s of temporal constraints will be established so as to check the constraints automatically by the model checking tool UPPAAL. In this section, we propose the procedure for locating temporal constraints, as depicted in Fig. 10.

For the sake of simplicity, in this paper we adopt the format $D_R(t_i, t_j) \leq s$ to describe temporal constraints (Du & Fan, 2010; Du et al., 2015a; Du et al., 2011; Du et al., 2015b; Du et al., 2013), which denote that t_j should end no later than s time units after t_i starts. If the time interval between the enabling time of t_i and possible firing time of t_j is less than or equal to s in all execution cases, then $D_R(t_i, t_j) \leq s$ is satisfied. Otherwise, it is violated (Du & Fan, 2010; Du et al., 2011; Du et al., 2015b; Du et al., 2013). For example, there is a temporal constraint $D_R(t_4, t_7) \leq 45$ in Table 4. If the time interval between the enabling time of t_4 and the possible firing time of t_7 is less than or equal to 45, then $D_R(t_4, t_7) \leq 45$ is satisfied. Otherwise, it is violated.

To check temporal constraints in *TAN*, the starting and ending transitions of a temporal constraint should be transformed to arcs in the *TAN*. Then we set the *send* channels in the arcs of *TAN* and add *wait* channels in the arc of the observing *TA*s, and treat the temporal constraints as guard conditions for observing *TA*s to check them efficiently. Furthermore, the starting and ending transitions of temporal constraints which have or do not have exchanged messages in the fragments can be checked.

Definition 6. (*Transformed temporal constraint*). The format of temporal constraint $D_R(t_i, t_j) \leq s$ is transformed to arcs in the *TAN*, which can be denoted as a three-tuple $TTC = \{ Arcs_i, Arcs_j, s \}$, where $Arcs_i = \{ < state_a, state_b >, ..., < state_c, state_d > \}$ are the arcs from $state_a$ to $state_b$, ..., and $state_c$ to $state_d$ in *TAN* corresponding to the starting transition of a temporal constraint in *ToN*, $Arcs_j = \{ < state_i, state_j >, ..., < state_m, state_n > \}$ are the arcs from $state_i$ to $state_j$, ..., $state_m$ to $state_n$ corresponding to its ending transition, and s is the value of temporal constraint.

Definition 7. (*Interactive transition*). An interactive transition is a type of transition that contains input or output messages. Specifically, a transition t is called an interactive transition if: $\exists$ $p_{in} \in P_I$ (or $p_{out} \in P_O$), $p_{in} \in {}^\bullet t$ (or $p_{out} \in t^\bullet$), where P_I is the set of input message places and P_O is the set of output message places.

Definition 8. (*Inner transition*). An inner transition is a type of transition that does not involve input or output messages. Specifically, a transition t is called an inner transition if: $\nexists$ $p_{in} \in P_I$ and $p_{out} \in P_O$, and $p_{in} \in {}^\bullet t$ and $p_{out} \in t^\bullet$, where P_I is the set of input message places and P_O is the set of output message places.

For the interactive transitions of temporal constraints, we present Algorithm 4 to locate their arcs in *TAN* so that we can obtain the transformed temporal constraints.

Algorithm 4: Locate a temporal constraint whose transitions are interactive transitions.

```
1  Input: TAN and temporal constraint D_R(t_i, t_j) ≤ φ_i;
2  Output: Transformed temporal constraint TTC_i = { Arcs_ij, Arcs_ik, s };
3  Initialization: Arcs_ij = ∅, Arcs_ik = ∅, and s = ∅;
4  for i = 1; i ≤ TAN.arc.length; i++ do
5      if state i and arc i in TAN are transformed from path && t_i, which is the first transition of path then
6          Obtain the arcs x, y, ..., z immediately before state i;
7          Set Arcs_ij = Arcs_ik ∪ {arc i, x, y, ..., z };
8  for j = 1; j ≤ TAN.arc.length; j++ do
9      if state j and arc j in TAN are transformed from path && t_j, which is the last transition of path then
10         Obtain the arcs o, p, ..., q immediately after state j;
11         Set Arcs_ik = Arcs_ik ∪ {arc j, o, p, ..., q };
12 s = φ_i.
```

Algorithm 4 describes the procedure for locating a temporal constraint whose transitions are interactive transitions. It involves three aspects: First, the algorithm initializes the transformed temporal constraint (*L*3). Second, the algorithm returns the arcs for the starting transition by determining whether the starting transition is the first transition of a path and *state i* and arc *i* in *TAN* transformed from the path (*L*4 ∼ *L*7). Then, it returns the arcs for the ending transition by determining whether the ending transition is the last transition of a path and *state j* and arc *j* in *TAN* transformed from the path (*L*8 ∼ *L*11). Finally, the algorithm sets s in the transformed temporal constraint as the value of temporal constraint (*L*12).

By assuming that there are m arcs for t_i and n arcs for t_j in a *TAN*, the time complexity of Algorithm 4 is $O(mn)$.

Considering the temporal constraints ① − ③ presented in Table 4, we can locate the arcs in the *TAN* based on Algorithm 4 and obtain the transformed temporal constraints { { $< state_{1-1}, state_{1-2} >$ }, { $< state_{1-3}, state_{1-4} >$ }, 45}, { { $< initial\ state_2, state_{2-1} >$ }, { $< state_{2-5}, state_{2-6} >$, $< state_{2-5}, state_{2-7} >$ }, 10}, and { { $< state_{2-5}, state_{2-6} >$, $< state_{2-5}, state_{2-7} >$ }, { $< state_{1-4}, initial\ state_1 >$ }, 12}. Temporal constraint ④ cannot be located in the *TAN* based on Algorithm 4 because the starting transition t_5 is an inner transition, so it does not have input or output messages.

To locate the temporal constraints whose relevant transitions such as t_x are inner transitions, we first obtain the interactive transition t_y before t_x in the *ToN* that has interactive messages and its arc in *TAN*, and then calculate the time interval between them. That is to say, assume that transition t_b is an inner transition in temporal constraint $D_R(t_a, t_b) \leq s$. We can find another transition t_c before t_b in *ToN*, where t_c has interactive messages, and extract the corresponding arcs of t_c in *TAN* by using Algorithm 4, namely the arcs of t_b. Then we need to calculate the time interval between t_b and t_c, denoted as D_{bc} using formula (1) to check whether $D_R(t_a, t_b) \leq s$.

$$D_{bc} = \sum_{i \in ST} x_i + \sum_{j \in CT} \alpha y_j \tag{1}$$

Here, x_i and y_j correspond to the durations of the transitions in the *ToN*; *ST* corresponds to the set of sequence transitions, *CT* corresponds to the set of choice transitions, and α=1 or 0 represents whether a transition is executed or not in a chosen structure of *ToN*.

Then, based on the above discussion, we propose Algorithm 5 to locate the temporal constraints whose transitions are inner transitions.

Algorithm 5: Locate a temporal constraint whose transitions are inner transitions.

1 **Input**: *TAN* and temporal constraint $D_R(t_i, t_j) \le \varphi_j$;
2 **Output**: Transformed temporal constraint $TTC_j = \{ Arcs_{jm}, Arcs_{jn}, s \}$;
3 Initialization: $Arcs_{jm} = \emptyset$, $Arcs_{jn} = \emptyset$, and $s = \varphi_i$;
4 **if** t_i has interactive messages **then**
5 Obtain $Arcs_{jm} = \{ < state_a, state_b >, ..., < state_c, state_d > \}$ for t_i based on Algorithm 4;
6 **else**
7 Find transition a t_y before t_i, where t_y has an interactive message;
8 Extract all the $state_i$ and arc_j in *TAN* for t_{iy} and the arcs immediately before *state i*, and insert them in $Arcs_{jm} = \{ < state_a, state_b >, ..., < state_c, state_d > \}$, calculate the time interval D_{yi} based on formula (1), and set $s = s + D_{yi}$;
9 **if** t_j has interactive messages **then**
10 Obtain $Arcs_{jn} = \{ < state_i, state_j >, ..., < state_m, state_n > \}$ for t_j based on Algorithm 4;
11 **else**
12 Find a transition t_z before t_j, where t_z has an interactive message;
13 Obtain all the $state_m$ and arc n in *TAN* for t_z and the arcs immediately after $state_m$, and insert them in $Arcs_{jn} = \{ < state_i, state_j >, ..., < state_m, state_n > \}$, calculate the time interval D_{zj} based on formula (1), and set $s = s + D_{zj}$.

Algorithm 5 describes the procedure for locating a temporal constraint whose transitions are inner transitions. It contains three steps: First, the algorithm sets *s* in the transformed temporal constraint as the value of the temporal constraint (*L*3). Then, the algorithm returns the arcs for the starting transition if it has interactive messages (*L*4 ∼ *L*5), or finds another transition t_y before t_i, where t_y has an interactive message, to obtain the arcs for the starting transition (*L*6 ∼ *L*8). Finally, the algorithm returns the arcs for the ending transition if it has interactive messages (*L*9 ∼ *L*10), or finds another transition t_z before t_j, where t_z has an interactive message, to obtain the arcs for the ending transition (*L*11 ∼ *L*13).

By assuming that there are *m* arcs for t_i and *n* arcs for t_j in a *TAN*, the time complexity of Algorithm 5 is $O(mn)$.

Reconsider the temporal constraint ④. We first find interactive transition t_4 before t_5. Then we calculate the time interval D_{45}=2 and obtain its transformed temporal constraint { { $< state_{1\text{-}1}, state_{1\text{-}2} >$ }, { $< state_{1\text{-}3}, state_{1\text{-}4} >$ }, 30}. However, we cannot transform this temporal constraint if we are using the *Partition – oriented method*. This is because the starting transition t_5 is an inner transition which has no input or output messages.

5.2. Checking temporal constraints by observing TAs

Based on Algorithms 4 and 5, we can locate the temporal constraints whose activities have or do not have exchanged messages in *TAN*, and we present the procedure for automatically constructing observing *TA*s as shown in Fig. 11. The detailed procedure for constructing observing *TA*s is presented in Algorithm 6 .

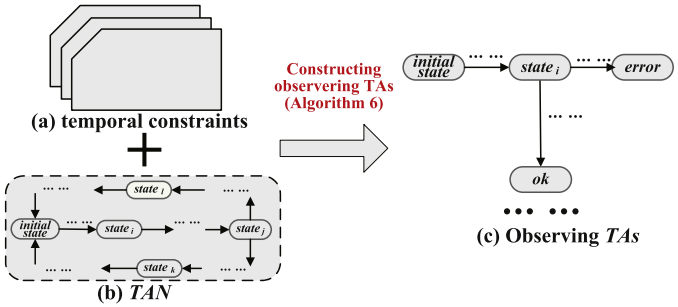


Fig. 11. Constructing observing *TA*s.

Algorithm 6: Construct observing *TA*s for temporal constraints.

1 **Input**: *TAN* and the set of temporal constraints *TC*;
2 **Output**: The set of observing *TA*;
3 Initialization: $TC = \{ tc_1, tc_2, ..., tc_n \}$, the set of observing *TA*s, i.e., $OT = \emptyset$;
4 **for** i=1; $i \le n$; i++ **do**
5 **if** t_{i1} and t_{i2} of tc_i have interactive messages **then**
6 Obtain $TTC = \{ Arcs_i, Arcs_j, s \}$ based on Algorithm 4;
7 **else**
8 Obtain $TTC = \{ Arcs_i, Arcs_j, s \}$ based on Algorithm 5;
9 Define a *TA* template Observer *i* with four locations: *initial state*, $state_i$, *state error*, and *state ok*, add the local clock *z* in the arc from the initial state to *state i*, and the guard condition $z > s$ in the arc from *state i* to *state error*;
10 Based on $Arcs_i = \{ < state_a, state_b >, ..., < state_c, state_d > \}$ in TTC_i, add the *send pab*, ..., *send pbc* in the arcs from *state a* to *state b*,..., *state c* and *state d* in *TAN* and the channels *wait pab*, ..., *wait pbc* in the arc from *initial state* to $state_i$ in *Observer i*;
11 According to $Arcs_j = \{ < state_i, state_j >, ..., < state_m, state_n > \}$ in TTC_i, add the *send pij*, ..., *send pmn* in the arcs from $state_i$ to $state_j$, ..., $state_m$ to $state_n$ in *TAN* and the channels *wait pij*, ..., *wait pmn* in the arc from $state_i$ to *state ok* in *Observer i*;
12 $OT = OT \cup \{ Observer\ i \}$.

Algorithm 6 depicts how to construct observing *TA*s for temporal constraints. The procedure for constructing observing *TA*s is as follows: First, the algorithm obtains the transformed temporal constraint by determining whether transitions have or do not have interactive messages (*L*4 ∼ *L*8). Then the algorithm constructs the structure information for an observing *TA* (*L*9) and adds the *send, wait* channels in the observing *TA* (*L*10 ∼ *L*11). Finally, it returns all of the observing *TA*s (*L*12).

By assuming that there are *n* temporal constraints in the models, the time complexity of Algorithm 6 is $O(n)$.

To confirm we can construct observing *TA* for any temporal constraint using Algorithm 6, the following theorem is proved.

Theorem 2. *Any temporal constraint can obtain its observing TA using Algorithm 6.*

Proof. Assume that there is a temporal constraint tc_i which cannot construct its observing *TA* by Algorithm 6. Each observing *TA* has default settings, namely four locations: *initial state*, $state_i$, *state error*, and *state ok*, in which clock *z* is labeled in the arc from *ini-*

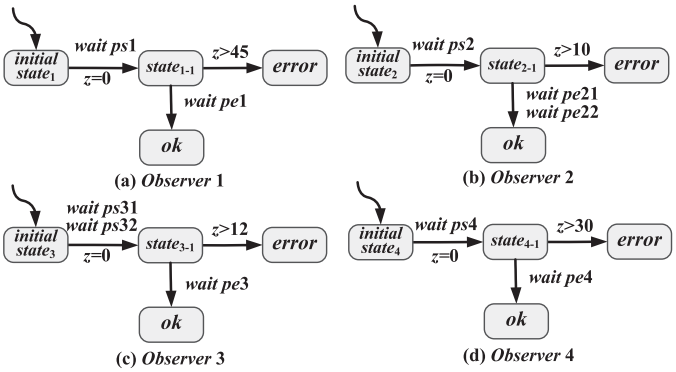


Fig. 12. *Observers* 1–4.

tial state to $state_i$, guard condition $z > s$ is added in the arc from $state_i$ to *state error*, *wait* channels are labeled in the arc from *initial state* to $state_i$ in the observing *TA* to start the observing *TA*, and *send* channels in the arc will from $state_i$ to *state ok* in *i* to end the observing *TA*.

Because locations and clock are fixed for each observing *TA*, the assumption means that one or more channels as well as guard conditions for tc_i do not exist. However, all the channels of tc_i can be extracted by Algorithm 4 or Algorithm 5 and the guard condition is the value of tc_i. This contradicts the assumption. Therefore, we conclude that any temporal constraint can obtain its observing *TA* using Algorithm 6. □

According to Algorithm 6, we can construct observing *TA*s for any temporal constraint. Then, based on the observing *TA*s and *TAN*, we can efficiently check the temporal constraints through queries in UPPAAL. The queries for checking the temporal constraints by UPPAAL are as follows: $f ::= A[]\ p \mid E <> p$; $p ::= a \mid p \mid (p) \mid p \vee p \mid p \wedge p \mid p \rightarrow p$, where a can denote a clock variable, an integer variable expression, or a position in a *TA*; $E <> p$ means it can reach a certain position (if it satisfies property p) from the starting point, and $A[]\ p$ indicates that all positions satisfy the property p.

Reconsider the temporal constraints ① – ④ in Table 4. Their transformed temporal constraints are { { $< state_{1-1}, state_{1-2} >$ }, { $< state_{1-3}, state_{1-4} >$ }, 45}, { { $< initial\ state_2, state_{2-1} >$ }, { $< state_{2-5}, state_{2-6} >$, $< state_{2-5}, state_{2-7} >$ }, 10}, { { $< state_{2-5}, state_{2-6} >$, $< state_{2-5}, state_{2-7} >$ }, { $< state_{1-4}, initial\ state_1 >$ }, 12}, and { { $< state_{1-1}, state_{1-2} >$ }, { $< state_{1-3}, state_{1-4} >$ }, 30}.

Then, we construct the observing *TA*s based on Algorithm 6. We take the temporal constraint ① as an example to illustrate Algorithm 6. Based on the transformed temporal constraint { { $< state_{1-1}, state_{1-2} >$ }, { $< state_{1-3}, state_{1-4} >$ }, 45}, we define a *TA* template *Observer* 1 with *initial* $state_1$, $state_{1-1}$, *state error*, *state ok* and the arcs between them and add the clock z=0 in the arc from *initial* $state_1$ to $state_{1-1}$ and guard condition $z > 45$ in the arc from $state_{1-1}$ to *state error*, as shown in Fig. 12(a). Then, we set *send ps*1 in the arc from $state_{1-1}$ to $state_{1-2}$ in Fig. 9(a), and set *wait ps*1 in the arc from *initial* $state_1$ to $state_{1-1}$ in Fig. 12(a). Finally, we add *send pe*1 in the arc from $state_{1-3}$ to $state_{1-4}$ in Fig. 9(a) and *wait pe*1 from $state_{1-1}$ to *state ok* in Fig. 12(a) so that we can construct *Observer* 1. In the same way, we can construct *Observers* 2, 3 and 4 for temporal constraints ② – ④, as shown in Figs. 12(b)–(d). Finally, we input the above observing *TA*s into UPPAAL and check them automatically; the results are shown in Table 7.

Because temporal constraints ② and ④ are violated, the *MToN* of *OSS* and *CS* is not time-compatible. The result shows that services *OSS* and *CS* cannot be composed to collaborate with each other. It is of great assistance for guiding users to reselect services or modify the inner structure of services *OSS* and *CS*.

Table 7
Description of temporal constraints.

No.	Observing *TA*s	Queries	Checking results
①	Observer 1	*A*[] not *Observer* 1.*error*	Satisfied
②	Observer 2	*A*[] not *Observer* 2.*error*	Violated
③	Observer 3	*A*[] not *Observer* 3.*error*	Satisfied
④	Observer 4	*A*[] not *Observer* 4.*error*	Violated

Although the *Transition – oriented method* can be used to check the temporal constraints ① – ④, its *TAN* model is large. This will lead to low efficiency in verification. If we adopt the *Partition – oriented method*, the temporal constraints in ④ cannot be checked directly.

Furthermore, there are several methods to update and modify services to re-satisfy temporal constraints (Du et al., 2011). If we want to satisfy temporal constraints ② and ④, we can reduce the execution time of transitions t_6, t_7, and t_9 from [4, 6], [20, 25], and [1, 3] to [2, 4], [15, 20], and [1, 2], respectively. A detailed explanation of updating and modifying services to re-satisfy temporal constraints (Du et al., 2011) is omitted, because it is beyond the scope of this paper.

6. Performance analysis

In order to analyze the performance of our approach quantitatively, we compare it with the *Transition – oriented method* and *Partition – oriented method* by the following Experiments 1, 2, and 3.

6.1. Experimental setup

Compared methods: (1) *Transition – oriented method*. This method constructs a *TA* for each activity, and checks temporal constraints by observing *TA*s. (2) *Partition – oriented method*. This method decomposes each service of composition models into several fragments, transforms fragments into a *TAN*, and then checks temporal constraints by observing *TA*s. (3) *Service – oriented method*. The approach proposed in this paper.

Because *Transition – oriented method* and *Partition – oriented method* cannot transform an *MToN* with complex mediation transitions to a *TAN*, they cannot be used in our experiments directly. Thus, the following setting is necessary when we compare their performances with our *Service – oriented method*.

Settings: The complex mediation transitions in an *MToN* are decomposed by our *Service – oriented method*. Then, we use the *Transition – oriented method*, *Partition – oriented method*, and *Service – oriented method* to construct the *TAN* for the decomposition model. Finally, we validate the temporal constraints by *TAN* and observing *TA*s through UPPAAL.

In all experiments, we vary the number of places/transitions randomly in the *ToN*s. All experiments were conducted on a desktop computer with a 2.6 GHz Intel Core *i*5 CPU and 4 GB RAM running the Windows 7 Operating System. The UPPAAL version is 4.1.13 Academic.

Criteria: We compare the performance of the three approaches according to the number of states and arcs in the *TAN* and the validation time of temporal constraint for the same *MToN*.

6.2. Evaluation using test cases

The detailed information and the results of the three experiments follow.

Table 8
Comparison for Experiment 1.

No.	Test case	*Transition-oriented method*		*Partition-oriented method*		*Service-oriented method*	
1	2(12,6,2)	16	16	11	12	7	8
2	2(16,11,2)	26	26	9	11	8	10
3	2(20,12,3)	30	30	18	20	9	10
4	2(23,17,3)	40	40	17	19	11	14
5	2(41,23,6)	58	58	31	38	13	13
6	2(42,33,7)	80	80	39	47	19	27
7	2(59,37,8)	90	90	47	58	17	20
8	2(65,53,7)	120	120	42	58	25	34
9	2(70,57,9)	132	132	58	71	28	37
10	2(92,67,11)	156	156	62	79	32	43

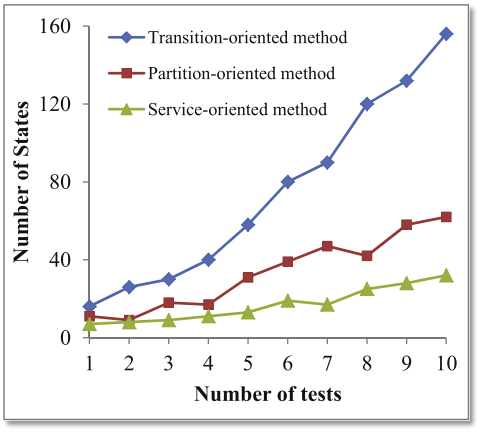


Fig. 13. Increasing trend of states in Experiment 1.

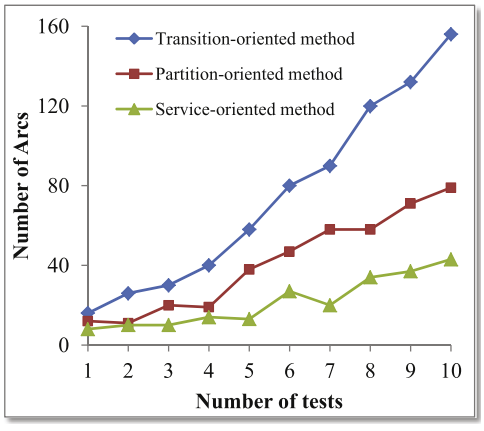


Fig. 14. Increasing trend of arcs in Experiment 1.

Experiment 1: Changing the number of places/transitions in the *MToN* with a fixed number of *ToNs*.

In the test cases, we establish ten test cases involving two *ToNs* and vary the numbers of places/transitions from 18 to 159 with random structures of *ToNs*. The experimental results are shown in Table 8, and further depicted in Figs. 13 and 14.

In Table 8, the first column indicates the case number and the second implies the test case using the form "$n(a, b, c)$" where n means n *ToNs*, a, b, and c represent a places, b transitions, and c mediation transitions; for example, 2(12, 6, 2) means 2 *ToNs* with 12 places, 6 transitions, and 2 mediation transitions. The third, fifth, and seventh columns indicate the number of states in *Transition – oriented method*, *Partition – oriented method*, and *Service – oriented method*, respectively. The fourth, sixth, and eighth columns indicate the number of corresponding arcs, respectively.

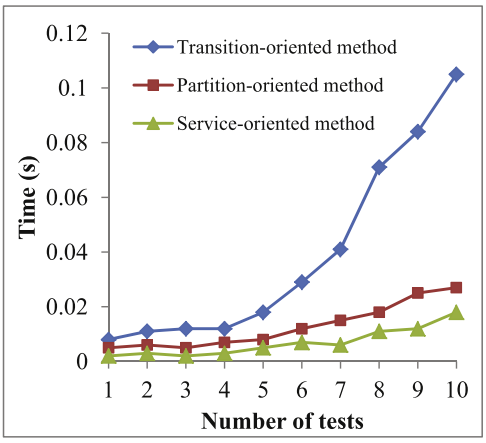


Fig. 15. Validation time of temporal constraint for each method in Experiment 1.

The results of Experiment 1 are shown in Table 8, Figs. 13, and 14. The results show that the *TA* models constructed by our approach are much more compact than the ones obtained by the *Transition – oriented method* and *Partition – oriented method*. Specifically, (1) for the number of states in the *TA* model, the maximum, minimum, and average improvements of effectiveness in our method compared to the *Transition – oriented method*, are 81.1%, 56.1%, and 74%, respectively, and are 63.8%, 11.1%, and 44.7%, respectively, compared to the *Partition – oriented method*; (2) for the number of arcs in the *TA* model, the maximum, minimum, and average improvements of effectiveness in ours compared to the *Transition – oriented method*, are 77.8%, 61.5%, and 68.1%, respectively, and are 65.8%, 10.1%, and 42.7%, respectively, compared to the *Partition – oriented method*. Given the presented experimental results, we can conclude that our approach dramatically decreases the complexity of *TA* in contrast to the *Transition – oriented method* and *Partition – oriented method* in Experiment 1.

Next, we set a temporal constraint randomly for each test case in Table 8 and verify them by UPPAAL based on the above three approaches, respectively. The verification time (Time (s)) of each temporal constraint is shown in Table 9, and is depicted in Fig. 15.

Table 9 and Fig. 15 indicate that our method is much more economic than the others in the verification time of temporal constraints. Specifically, the maximum, minimum, and average improvements of effectiveness in our method compared to the *Transition – oriented method* are 85.7%, 72.2%, and 78.9%, respectively, and are 60%, 37.5%, and 50.1%, respectively, compared to the *Partition – oriented method*. Given the presented experimental results, we can conclude that the verification time required by our approach is much lower than the verification time of the *Transition – oriented method* and *Partition – oriented method* in Experiment 1.

Experiment 2: Changing the number of *ToNs* in the *MToN*

We establish ten test cases with increasing numbers of *ToNs* (from 2 to 11) and random quantities of places/transitions in these *ToNs*. Table 10 shows the experimental results, which are depicted in Figs. 16 and 17.

Table 10, Figs. 16, and 17 illustrate the results of Experiment 2. Specifically, the results show that, for different sizes of mediation-aided service composition models (i.e., different mediation-aided service composition models with different services and interactive messages), the *TA* models constructed by our approach are much more compact than the ones obtained by the *Transition – oriented method* and *Partition – oriented method*. Specifically, (1) for the number of states in the *TA* model, the maximum,

Table 9
Comparison of validation times for Experiment 1.

No.	Test case	Transition-oriented method time (s)	Partition-oriented method time (s)	Service-oriented method time (s)
1	2(12,6,2)	0.008	0.005	0.002
2	2(16,11,2)	0.011	0.006	0.003
3	2(20,12,3)	0.012	0.005	0.002
4	2(23,17,3)	0.012	0.007	0.003
5	2(41,23,6)	0.018	0.008	0.005
6	2(42,33,7)	0.029	0.012	0.007
7	2(59,37,8)	0.041	0.015	0.006
8	2(65,53,7)	0.071	0.018	0.011
9	2(70,57,9)	0.084	0.025	0.012
10	2(92,67,11)	0.105	0.027	0.018

Table 10
Comparison for Experiment 2.

No.	Test case	Transition-oriented method		Partition-oriented method		Service-oriented method	
1	2(12,7,2)	18	18	10	12	6	7
2	3(19,10,3)	26	26	18	19	11	12
3	4(28,18,4)	44	44	22	24	13	15
4	5(41,26,5)	62	62	30	31	17	20
5	6(61,39,7)	92	92	43	52	22	25
6	7(67,45,7)	104	104	48	59	27	33
7	8(72,46,9)	110	110	50	62	24	28
8	9(84,60,9)	138	138	54	65	27	35
9	10(94,62,10)	144	144	66	76	31	39
10	11(104,70,10)	160	160	77	87	34	40

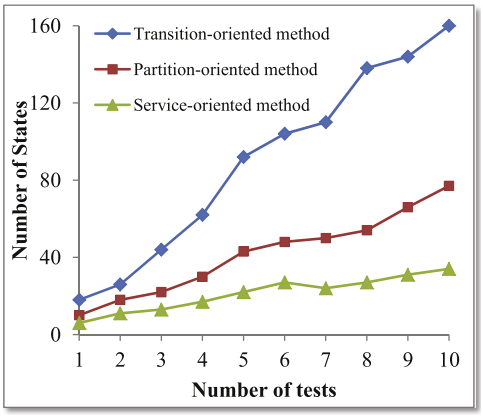


Fig. 16. Increasing trend of states in Experiment 2.

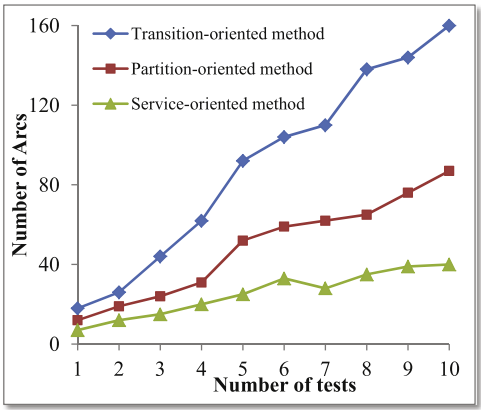


Fig. 17. Increasing trend of arcs in Experiment 2.

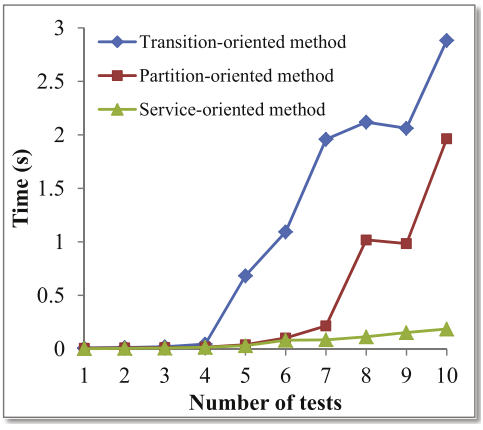


Fig. 18. Validation time of temporal constraint for each method in Experiment 2.

minimum, and average improvements of effectiveness in our method, compared to the *Transition – oriented method*, are 80.4%, 57.7%, and 72.7%, respectively, and are 53%, 38.9%, and 45.6%, respectively, compared to the *Partition – oriented method*; (2) for the number of arcs in the *TA* model, the maximum, minimum, and average improvements of effectiveness in ours, compared to the *Transition – oriented method*, are 74.6%, 53.8%, and 68%, respectively, and are 54.9%, 35.5%, and 44.1%, respectively, compared to the *Partition – oriented method*. Given the presented experimental results, we can conclude that our approach dramatically decreases the complexity of the *TAN* in contrast to the *Transition – oriented method* and *Partition – oriented method* in Experiment 2.

Next, we randomly set a temporal constraint for each test case in Table 10 and verify them using UPPAAL based on the *Transition – oriented method* and *Partition – oriented method* and our approach. The verification time (Time (s)) for each temporal constraint is shown in Table 11, which is further depicted in Fig. 18.

Table 11
Comparison of validation times for Experiment 2.

No.	Test case	Transition-oriented method time (s)	Partition-oriented method time (s)	Service-oriented method time (s)
1	2(12,7,2)	0.007	0.005	0.001
2	3(19,10,3)	0.012	0.008	0.004
3	4(28,18,4)	0.019	0.011	0.007
4	5(41,26,5)	0.044	0.015	0.014
5	6(61,39,7)	0.683	0.038	0.029
6	7(67,45,7)	1.093	0.101	0.185
7	8(72,46,9)	1.958	0.214	0.084
8	9(84,60,9)	2.119	1.018	0.112
9	10(94,62,10)	2.061	0.983	0.152
10	11(104,70,10)	2.883	1.963	0.185

Table 12
Comparisons for Experiment 3.

No.	Test case	Transition-oriented method		Partition-oriented method		Service-oriented method	
1	2(42,33,7)	80	80	39	47	19	27
2	2(47,33,9)	84	84	42	50	21	29
3	2(51,33,11)	88	88	43	51	22	30
4	2(55,33,13)	92	92	46	53	24	32
5	2(59,33,15)	96	96	49	57	26	34
6	2(63,33,17)	100	100	53	61	29	36
7	2(67,33,19)	104	104	56	64	30	38
8	2(71,33,21)	108	108	59	65	32	40
9	2(75,33,23)	112	112	60	68	33	41
10	2(79,33,25)	116	116	61	69	35	42

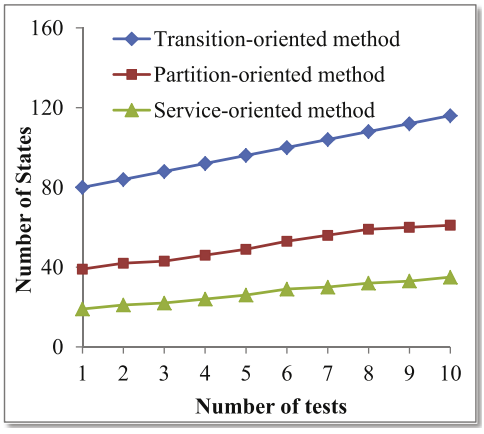


Fig. 19. Increasing trend of states in Experiment 3.

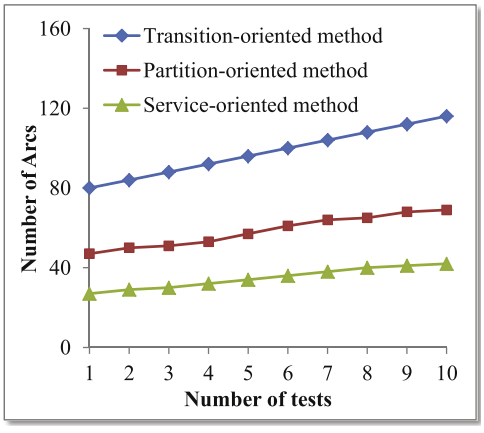


Fig. 20. Increasing trend of arcs in Experiment 3.

Table 11 and Fig. 18 demonstrate that our approach is much better than the others regarding the verification time of temporal constraints. The maximum, minimum, and average improvements of effectiveness in our method, compared to the *Transition – oriented method*, are 95.8%, 63.2%, and 84.9%, respectively, and are 90.6%, 20.1%, and 54%, respectively, compared to the *Partition – oriented method*. Given the presented experimental results, we can conclude that the verification time required by our approach is much less than the verification times for the *Transition – oriented method* and *Partition – oriented method* in Experiment 2.

Experiment 3: Changing the number of mediation transitions in *MToN* with a fixed number of *ToNs*.

In this experiment, we take the test case 2(42, 33, 7) from Table 8 in Experiment 1 and increase the number of mediation transitions. Table 12 shows the experimental results, which are depicted in Figs. 19 and 20.

The results of Experiment 3 are depicted in Table 12, Figs. 19, and 20. They show that the *TA* models constructed by our method are much more compact than the ones obtained by the *Transition – oriented method* and *Partition – oriented method*. Specifically, (1) for the number of states in the *TA* model, the maximum, minimum, and average improvements of effectiveness in our method compared to the *Transition – oriented method*, are 76.1%, 69.9%, and 72.1%, respectively, and are 51.3%, 42.7%, and 47%, respectively, compared to the *Partition – oriented method*; (2) for the number of arcs in the *TA* model, the maximum, minimum, and average improvements of effectiveness in ours compared to the *Transition – oriented method*, are 66%, 63%, and 64.5%, respectively, and are 42.6%, 38.5%, and 40.5%, respectively, compared to the *Partition – oriented method*. Given the presented experimental results, we can conclude that our approach dramatically decreases the complexity of the *TAN* in contrast to the *Transition – oriented method* and *Partition – oriented method* in Experiment 3.

Next, we randomly set a temporal constraint for each test case in Table 12, and verify them using UPPAAL based on *Transition – oriented method* and *Partition – oriented method* and our work. The

Table 13
Comparison of validation times for Experiment 3.

No.	Test case	Transition-oriented method time (s)	Partition-oriented method time (s)	Service-oriented method time (s)
1	2(42,33,7)	0.029	0.012	0.007
2	2(47,33,9)	0.044	0.013	0.008
3	2(51,33,11)	0.059	0.022	0.01
4	2(55,33,13)	0.066	0.029	0.009
5	2(59,33,15)	0.071	0.031	0.012
6	2(63,33,17)	0.081	0.036	0.013
7	2(67,33,19)	0.093	0.039	0.015
8	2(71,33,21)	0.097	0.042	0.017
9	2(75,33,23)	0.102	0.045	0.017
10	2(79,33,25)	0.109	0.056	0.019

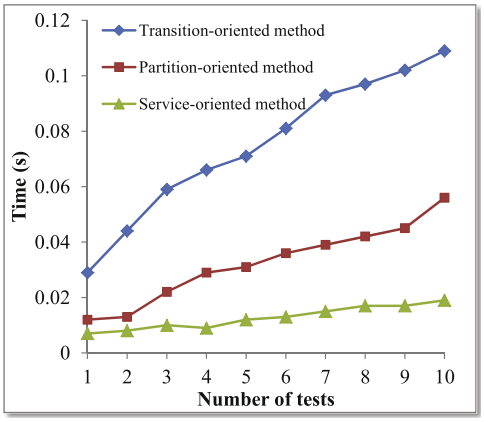


Fig. 21. Validation time of temporal constraint for each method in Experiment 3.

verification time (Time (s)) of each temporal constraint is shown in Table 13, and is depicted in Fig. 21.

Table 13 and Fig. 21 indicate that our approach is much more economic than the others regarding the verification time of temporal constraints. The maximum, minimum, and average improvements of effectiveness in our method compared to the *Transition – oriented method*, are 86.4%, 75.9%, and 82.6%, respectively, and are 66.1%, 38.5%, and 57.8%, respectively, compared to the *Partition – oriented method*. Given the presented experimental results, we can conclude that the required verification time of our approach is much lower than the verification times of the *Transition – oriented method* and *Partition – oriented method* in Experiment 3.

Conclusion of Experiments 1–3: Given the experimental results presented in Tables 8–13 and Figs. 13–21, we conclude that compared with the *Transition – oriented method* and *Partition – oriented method*, ours is more effective and efficient with regard to the states/arcs in *TAN* and required validation time of temporal constraints for the same *ToN*.

7. Related work

In this section, we compare our approach with existing approaches in four categories, i.e., mediation-aided composition of services, analyzing temporal constraints, model checking for temporal constraints, and fragmentation of Petri net models. Finally, we present a summary table to illustrate the comparison between our approach and existing ones.

7.1. Mediation-aided composition of services

Usually, two (or more) services providing complementary functionality could be linked together in principle, but cannot be directly composed because of partially compatible interfaces or interaction patterns. To solve this significant problem, mediation-aided composition based approaches have been proposed (Du et al., 2012; Li et al., 2010; Mrissa et al., 2013; Tan et al., 2009). Tan et al. (2009) analyzed compatibility of services by adopting mediation as a lightweight approach to make them compatible without changing their internal logic. Li et al. (2010) proposed a mediator pattern-based approach to generate executable mediators and glue partially compatible services together. Du et al. (2012) presented a Petri net approach to composing services using mediation transitions. Mrissa et al. (2013) proposed a context-based mediation approach to solve semantic heterogeneities between composed Web services.

The existing works (Du et al., 2012; Li et al., 2010; Mrissa et al., 2013; Tan et al., 2009) can address the issue of mediation-aided service composition by using a set of mediators/adaptors to glue two or more partially compatible services, but they do not fully investigate the time information of services such that they cannot analyze timed compatibility for mediation-aided service composition.

7.2. Analyzing temporal constraints

Temporal constraints analysis is a classic topic in the field of workflow and service composition. Important aspects of time management include estimating workflow/services execution duration, avoiding deadline violations, and satisfying all external time constraints such as fixed-date constraints and upper and lower bounds for time intervals between activities. In this section, we analytically describe the advances over the state of the art from two aspects: analyzing temporal constraints of workflow and analyzing temporal constraints of service composition.

7.2.1. Analyzing temporal constraints of workflow

Liu et al. (2011a), Liu et al. (2011b), and Liu et al. (2014) first assign a set of point selection strategies for temporal violation handling by analyzing the throughput of the overall workflow execution with frequent temporal violations, and then prevent the temporal violation based on strategy. They focus on preventing future potential temporal violations once they are detected. Han, Liu, Wen, and Wang (2010) presented a dynamic approach to analyzing time constraints to assist managers in solving time constraint violations. Ma et al. (2014) proposed a graph distance based approach for measuring the similarity between data oriented workflows with variable time constraints.

This type of method concentrates on the temporal analysis of activities in a single process; they consider neither data formats of the exchanged messages nor their sequence consistency for multiple services/processes.

7.2.2. Analyzing temporal constraints of service composition

Leal, Bonillo, and Rey (2009) described causal temporal constraint networks (CTCN) as a new computable model for representing temporal information and efficiently handling causality. Santos et al. (2012) proposed a framework to reason over uncertainty under temporal constraints. Guermouche and Godart (2010) analyzed the interoperability of Web services by data and timed constraints. Gao and Singh (2014) extracted business events and their temporal constraints from contract text, and used topic modeling to automatically organize the event terms into clusters. These methods can be used to check the temporal constraints, but they cannot analyze the timed compatibility for mediation-aided service composition, as they do not fully investigate the message mismatches between services in a composition.

Du et al. (2015a) and Du and Zhang (2014) proposed new approaches to check the temporal constraints of mediation-aided service composition. Xing et al. (2016) proposed a new approach to modeling and analyzing timed compatibility of service composition where time durations are mixture distributed. They can be used to analyze the timed compatibility for mediation-aided services composition; however, they suffer from low efficiency when checking the temporal constraints of mediation-aided service composition because the number of system states exponentially increases with the number and complexity of services. (Du, Yu, Yang, & Wang, 2016) aimed at describing the relationships among activities in service processes based on a declarative service flow language, and then constructed an improved modular timed state graph to analyze the temporal constraints for mediation-aided composition of service processes with a relation network. In this paper, we focus on analyzing timed compatibility of mediation-aided Web service composition efficiently via model checking. Our approach treats each service in the mediation-aided Web service composition as a fragment, transforms fragments into *TAN*, and checks all types of temporal constraints. The approach presented in Du et al. (2016) can analyze the temporal constraints for mediation-aided composition of service processes with a relation network, but it cannot deal with the problem we focus on in this paper. This is because it can neither fragment the mediation-aided Web service composition, nor transform mediation-aided Web service composition into a *TAN* to check temporal constraints efficiently.

7.3. Model checking for temporal constraints

Model checking is a well-established technique to automatically verify complex systems. It allows us to determine whether or not a system model meets a specification. Based on model checking, we not only can verify specifications for languages or protocols, but can also check temporal constraints for workflow/service composition.

7.3.1. Checking timed requirements for languages or protocols

Existing works (Baouya, Bennouar, Mohamed, & Ouchani, 2015; Bataineh et al., 2017; Bentahar et al., 2013; Fernndez, Menndez, & Camacho, 2017; Guermouche & Godart, 2009; Kokash et al., 2010; Ponge et al., 2010; Su, Rosenblum, & Tamburrlli, 2016) can be classified into two categories: checking timed requirement for languages and checking timed requirement for protocols. The former category (Baouya et al., 2015; Bataineh et al., 2017; Kokash et al., 2010; Su et al., 2016) addresses the issue of verifying whether a timed requirement in the target model expressed in the PRISM language meets all requirements. The latter category (Bentahar et al., 2013; Fernndez et al., 2017; Guermouche & Godart, 2009; Ponge et al., 2010) focuses on investigating a model checking based approach that involves checking some desirable properties among timed protocols.

Existing research works focus on analysis and checking of timed requirements for languages or protocols, such as BPEL and PLC programs, but they are unable to transform the structural models of mediation-aided service composition to a *TAN*, because they do not consider structure information involving workflow or service.

7.3.2. Checking temporal constraints for Petri net models

This type of research work can be divided into two methods: *Transition − oriented method* and *Partition − oriented method*.

1) *Transition − oriented method* (Du & Fan, 2010; Du et al., 2013; Gu & Kang, 2002). This method constructs a corresponding *TA* for each transition (activity) to analyze temporal constraints or other constraints in the target model.

2) *Partition − oriented method* (Du et al., 2015b). To improve the efficiency in checking temporal constraints, the services are decomposed into fragments, and then they are transformed into a *TAN* so that timed compatibility can be checked with fewer *TAs*.

The existing works suffer from the following disadvantages: (1) they are difficult to modify and maintain in a real-world business environment, and any complex mediation transitions cannot be decomposed; (2) *TANs* contain redundant states and arcs which may generate a large *TAN*, leading to low efficiency; and (3) the temporal constraints whose activities do not have exchanged messages cannot be located in the *TAN* such that observing *TAs* cannot be automatically constructed to check them efficiently.

7.4. Fragmentation of Petri net models

Workflow/service composition model fragmentation is used to partition a workflow/service composition model into fragments. Some exhaustive and approximate methods have been proposed (Kim, 2012; Sun, Zeng, & Wang, 2011; Tan & Fan, 2007). Tan and Fan (2007) proposed a novel dynamic workflow model fragmentation method to centralize a process model into fragments when the process is executed. Sun et al. (2011) proposed the application of process mining to discover workflow models and demonstrated how to determine the minimum time to complete a workflow and how to partition the workflow to achieve efficient server usage. Kim (2012) proposed a model-driven workflow fragmentation framework that provides a series of fragmentation algorithms to semantically fragment a workflow model.

The advantages of fragmentation methods include improved efficiency to complete a model, enhanced scalability to outsource business functionalities, increased flexibility to designate execution sites on-the-fly, and avoidance of redundant information transfer by judging pre-conditions before forwarding fragments. However, the existing research works (Kim, 2012; Sun et al., 2011; Tan & Fan, 2007) focus on partitioning a model, but they do not fully consider analyzing the timed compatibility for mediation-aided service composition, or suffer from low efficiency when services and activities are large in number.

7.5. Comparison of existing methods and our approach

In this subsection, we compare our approach with the existing approaches in terms of analyzing temporal constraints of workflow, service composition, and mediation-aided composition of services, as well as comparing whether they can or cannot transform mediation-aided composition of services with complex mediation transitions to *TAN*, construct a compact *TAN*, and check all types of temporal constraints.

A comparative summary table is given in Table 14. The columns of the table correspond to the following criteria, where "$\surd$" means a method can, whereas "$\times$" means it cannot perform a certain function.

Table 14
Comparison among existing research works and ours.

Related works	References	ATCW	ATCS	ATCM	TMT	CCT	AATC
Mediation-aided composition of services	Tan et al. (2009)	×	×	×	×	×	×
	Li et al. (2010)	×	×	×	×	×	×
	Du et al. (2012)	×	√	√	×	×	×
	Mrissa et al. (2013)	×	×	×	×	×	×
Analyzing temporal constraints	Liu et al. (2011a)	√	×	×	×	×	×
	Liu et al. (2011b)	√	×	×	×	×	×
	Liu et al. (2014)	√	×	×	×	×	×
	Han et al. (2010)	√	×	×	×	×	×
	Ma et al. (2014)	√	×	×	×	×	×
	Leal et al. (2009)	×	√	×	×	×	×
	Santos et al. (2012)	×	√	×	×	×	×
	Guermouche and Godart (2010)	×	√	×	×	×	×
	Gao and Singh (2014)	×	√	×	×	×	×
	Du et al. (2015a)	×	√	×	×	×	×
	Du and Zhang (2014)	×	√	×	×	×	×
	Xing et al. (2016)	×	√	×	×	×	×
Model checking of temporal constraints	Guermouche and Godart (2009)	×	×	×	×	×	×
	Ponge et al. (2010)	×	×	×	×	×	×
	Kokash et al. (2010)	×	×	×	×	×	×
	Bentahar et al. (2013)	×	×	×	×	×	×
	Baouya et al. (2015)	×	×	×	×	×	×
	Su et al. (2016)	×	×	×	×	×	×
	Bataineh et al. (2017)	×	×	×	×	×	×
	Fernndez et al. (2017)	×	×	×	×	×	×
	Gu and Kang (2002)	√	×	×	×	×	√
	Du and Fan (2010)	√	×	×	×	×	√
	Du et al. (2013)	√	×	×	×	×	√
	Du et al. (2015b)	√	√	×	×	√	×
Petri net models fragmentation	Tan and Fan (2007)	×	×	×	×	×	×
	Sun et al. (2011)	×	×	×	×	×	×
	Kim (2012)	×	×	×	×	×	×
Our approach		√	√	√	√	√	√

• **ATCW** means that a method can analyze temporal constraints of workflow.
• **ATCS** indicates that a method can analyze temporal constraints of service composition.
• **ATCM** implies that a method can analyze temporal constraints of mediation-aided composition of services.
• **TMT** means that a method can transform mediation-aided composition of services with complex mediation transitions to *TAN*.
• **CCT** indicates that a method can construct a compact *TAN* to efficiently analyze temporal constraints.
• **AATC** implies that a method can analyze all types of temporal constraints, including the ones whose activities have or do not have exchanged messages.

8. Conclusion

Analyzing the satisfaction of temporal constraints (i.e., timed compatibility) is a common and sometimes serious problem in real-life situations. Thereby, it is necessary to verify the timed compatibility of mediation-aided service composition. In this paper, we propose a new model checking approach to analyze timed compatibility, which is divided into three stages. First, mediation-aided service composition is decomposed into fragments based on services, each of which includes the service and its input and output messages. Second, each fragment is transformed into a *TA* based on two types of transformations: structure and interactive messages. Third, observing *TA*s are constructed for the temporal constraints whose activities have or do not have exchanged messages to automatically check them.

The theoretical contributions of our approach are as follows: (1) Mediation-aided service composition with complex mediation transitions is handled. This greatly expands the application scope of our approach, such that any service can be analyzed by enterprises. Services are subject to many changes, so this flexible and effective decomposition strategy can facilitate service changes in complex real-world situations. This is very important for enterprise management and decision-making in the current fiercely competitive business environment. (2) A compact *TAN* can be constructed for fragments in which the number of states and arcs is dramatically decreased and the verification time is significantly reduced compared with existing methods. This greatly improves the management efficiency of enterprises, saving them time and costs of decisions. (3) Temporal constraints whose activities have or do not have exchanged messages can be located in the *TAN*. This expands the verification capability to all temporal constraints, such that we can provide more powerful validation and analysis, and ensure that enterprises do not violate any law or regulation.

The drawbacks of our approach are as follows, and solving them can be viewed as our future research directions. (1) In real business environments, owing to the strategies, laws, or regulations, there are usually many positive or negative execution relationships among the services (the so called relation network). How to decompose a model which contains relation networks should be fully investigated in future work. (2) When system scale is very large, with too many domains, services, and interactive messages, its Petri net model may be very complex, such that the model checking based approach may experience the traditional space explosion limitation. Under this circumstance, how to conduct efficient interactive verification instead of re-constructing the full automaton model to efficiently verify temporal constraints should be considered. (3) In this paper, we assume that the time interval assigned to every activity is fixed. However, in a dynamic and complex business environment, these time intervals will change for multiple reasons. Future research may be to develop more elabo-

rate algorithms to include variable-length activities. (4) In this paper, one must notice that such a set of temporal constraints is not minimal. In other words, some temporal constraints may be redundant, implying that some temporal constraints are dependent on others. Therefore, it is desirable to identify a way to remove them while preserving the independent constraints.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant no. 61473035.

References

Baouya, A., Bennouar, D., Mohamed, A. O., & Ouchani, S. (2015). A quantitative verification framework of sysML activity diagrams under time constraints. *Expert Systems with Applications, 42*(21), 7493–7510.

Baruwa, O. T., Piera, M. A., & Guasch, A. (2015). Deadlock-free scheduling method for flexible manufacturing systems based on timed colored petri nets and anytime heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45*(5), 831–846.

Bataineh, A. S., Bentahar, J., Menshawy, M. E., & Dssouli, R. (2017). Specifying and verifying contract-driven service compositions using commitments and model checking. *Expert Systems with Applications, 74*, 151–184.

Bentahar, J., Yahyaoui, H., Kova, M., & Maamar, Z. (2013). Symbolic model checking composite web services using operational and control behaviors. *Expert Systems with Applications, 40*(2), 508–522.

Bernardi, S., & Campos, J. (2013). A min-max problem for the computation of the cycle time lower bound in interval-based time Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 43*(5), 1167–1181.

Chang, W. L., & Yuan, S. T. (2009). A Markov based collaborative pricing system for information goods bundling. *Expert Systems with Applications, 36*(2), 1660–1674.

Du, Y., & Fan, Y. (2010). Real-time model checking of dynamic temporal consistency for multi-process of workflow. *Journal of Mechanical Engineering, 46*(2), 186–191. (in Chinese version)

Du, Y., Li, X., & Xiong, P. (2012). A petri net approach to mediation-aided composition of web services. *IEEE Transactions on Automation Science and Engineering, 9*(2), 429–435.

Du, Y., Tan, W., & Zhou, M. C. (2015a). Timed compatibility analysis of web service composition: A modular approach based on petri nets. *IEEE Transactions on Automation Science and Engineering, 11*(2), 594–606.

Du, Y., Xiong, P., Fan, Y., & Li, X. (2011). Dynamic checking and solution to temporal violations in concurrent workflow processes. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, 41*(6), 1166–1181.

Du, Y., Yang, B., & Tan, W. (2015b). A model checking approach to analyzing timed compatibility in mediation-aided composition of web services. In *Proceedings of the IEEE international conference on web service* (pp. 567–574).

Du, Y., Yu, Z., Yang, B., & Wang, Y. (2016). Temporal consistency analysis of mediation-aided composition of service processes with relation network. In *Proceedings of the international conference on internet of things and IEEE green computing and communications and IEEE cyber, physical and social computing and IEEE smart data* (pp. 625–630).

Du, Y., & Zhang, W. (2014). An operating guideline based approach to analyzing timed compatibility of service composition. In *Proceedings of the international conference on e-business engineering* (pp. 131–138).

Du, Y., Zhang, W., & Tan, W. (2013). Pattern-based model checking for dynamic analysis of workflow processes with temporal constraints. In *Proceedings of the international conference on signal image technology & internet based systems* (pp. 225–232).

Fernndez, V. R., Menndez, H. D., & Camacho, D. (2017). Analysing temporal performance profiles of UAV operators using time series clustering. *Expert Systems with Applications, 70*, 103–118.

Gao, X., & Singh, M. P. (2014). Mining contracts for business events and temporal constraints in service engagements. *IEEE Transactions on Services Computing, 7*(3), 427–439.

Gu, Z. H., & Kang, G. S. (2002). Analysis of event-driven real-time systems with time petri nets: A translation-based approach. In *Proceedings of the IFIP 17th world computer congress* (pp. 31–40).

Guermouche, N., Perrin, O., & Ringeissen, C. (2008). A mediator based approach for services composition. In *Proceedings of International Conference on Software Engineering Research, Management and Applications* (pp. 273–280).

Guermouche, N., & Godart, C. (2009). Timed model checking based approach for web services analysis. In *Proceedings of the IEEE international conference on web services* (pp. 213–221).

Guermouche, N., & Godart, C. (2010). Timed conversational protocol based approach for web services analysis. In *Proceedings of the international conference on service-oriented computing* (pp. 603–611).

Han, R., Liu, Y., Wen, L., & Wang, J. (2010). Dynamically analyzing time constraints in workflow systems with fixed-date constraint. In *Proceedings of the Asia-pacific web conference* (pp. 99–105).

Hu, H., & Liu, Y. (2015). Supervisor synthesis and performance improvement in automated manufacturing systems by using petri nets. *IEEE Transactions on Industrial Informatics, 11*(2), 450–458.

Hu, H., Liu, Y., & Yuan, L. (2016). Supervisor simplifications in FMSs: Comparative studies and new results using petri nets. *IEEE Transactions on Control Systems Technology, 24*(1), 81–95.

Ibrahim, N., & Khalil, I. (2012). Verifying web services compositions using UPPAAL. In *Proceedings of the international conference on computer systems and industrial informatics* (pp. 1–5).

Jula, A., Sundararajan, E., & Othman, Z. (2014). Cloud computing service composition: A systematic literature review. *Expert Systems with Applications, 41*(8), 3809–3824.

Kholy, W. E., Bentahar, J., Menshawy, M. E., Qu, H., & Dssouli, R. (2014). Modeling and verifying choreographed multi-agent-based web service compositions regulated by commitment protocols. *Expert Systems with Applications, 41*(16), 7478–7494.

Kim, K. (2012). A model-driven workflow fragmentation framework for collaborative workflow architectures and systems. *Journal of Network and Computer Applications, 35*(1), 97–110.

Kokash, N., Krause, C., & deVink, E. P. (2010). Time and data aware analysis of graphical service models. In *Proceedings of the IEEE international conference on software engineering and formal methods* (pp. 125–134).

Leal, F., Bonillo, V. M., & Rey, E. M. (2009). Causal temporal constraint networks for representing temporal knowledge. *Expert Systems with Applications, 36*(1), 27–42.

Li, X., Fan, Y., Madnick, S., & Sheng, Q. Z. (2010). A pattern-based approach to protocol mediation for web service composition. *Information and Software Technology, 52*(3), 304–323.

Liu, X., Ni, Z., Chen, J., & Yang, Y. (2011a). A probabilistic strategy for temporal constraint management in scientific workflow systems. *Concurrency and Computation: Practice and Experience, 23*(16), 1893–1919.

Liu, X., Yang, Y., Jiang, Y., & Chen, J. (2011b). Preventing temporal violations in scientific workflows: Where and how. *IEEE Transactions on Software Engineering, 37*(6), 805–825.

Liu, X., Yang, Y., Yuan, D., & Chen, J. (2014). Do we need to handle every temporal violation in scientific workflow systems? *ACM Transactions on Software Engineering Methodology, 23*(1), 5:1–5:34.

Luo, J. C., Xing, K. Y., Zhou, M. C., Li, X. L., & Wang, X. N. (2015). Deadlock-free scheduling of automated manufacturing systems using petri nets and hybrid heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45*(3), 530–541.

Ma, Y., Zhang, X., & Lu, K. (2014). A graph distance based metric for data oriented workflow retrieval with variable time constraints. *Expert Systems with Applications, 41*(4), 1377–1388.

Mokadem, H., Berard, B., Gourcuff, V., Smet, O., & Roussel, J. (2010). Verification of a timed multitask system with uppaal. *IEEE Transactions on Automation Science and Engineering, 7*(4), 921–932.

Mrissa, M., Sellami, M., Vettor, P. D., Benslimane, D., & Defude, B. (2013). A decentralized mediation-as-a-service architecture for service composition. In *Proceedings of the IEEE international conference on enabling technologies: Infrastructure for collaborative enterprises* (pp. 80–85).

Ponge, J., Boualem, B., Casati, F., & Toumani, F. (2010). Analysis and applications of timed service protocols. *ACM Transactions on Software Engineering Methodology, 19*(4), 11:1–11:38.

Rezaei, R., Chiew, T. K., Lee, S. P., & Aliee, Z. S. (2014). Semantic interoperability framework for software as a service systems in cloud computing environments. *Expert Systems with Applications, 41*(13), 5751–5770.

Santos, E. Jr, Li, D., Santos, E. E., & Korah, J. (2012). Temporal bayesian knowledge bases c reasoning about uncertainty with temporal constraints. *Expert Systems with Applications, 39*(17), 12905–12917.

Su, G., Rosenblum, D., & Tamburrlli, G. (2016). Reliability of run-time quality-of-service evaluation using parametric model checking. In *Proceedings of the international conference on software engineering* (pp. 73–84).

Sun, S., Zeng, Q., & Wang, H. (2011). Process-mining-based workflow model fragmentation for distributed execution. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, 41*(2), 294–310.

Tan, W., & Fan, Y. (2007). Dynamic workflow model fragmentation for distributed execution. *Computers in Industry, 58*(5), 381–391.

Tan, W., Fan, Y., & Zhou, M. C. (2009). A petri net-based method for compatibility analysis and composition of web services in business process execution language. *IEEE Transactions on Automation Science and Engineering, 6*(1), 94–106.

Tan, W., Fan, Y., Zhou, M. C., & Tian, Z. (2010). Data-driven service composition in enterprise SOA solutions: A petri net approach. *IEEE Transactions on Automation Science and Engineering, 7*(3), 686–694.

Xing, Y., Li, S., Du, Y., & Liang, H. (2016). A new approach to modeling and analyzing timed compatibility of service composition under temporal constraints. In *Proceedings of the 15th international symposium on parallel and distributed computing* (pp. 306–313).