Verification of Composite Services with Temporal Consistency Checking and Temporal Satisfaction Estimation

Azlan Ismail, Jun Yan, and Jun Shen

School of Information Systems and Technology, University of Wollongong, Wollongong, NSW, 2522, Australia
{ai423, jyan, jshen}@uow.edu.au

Abstract. This paper aims to address the issue of consistency and satisfaction of composite services with the presence of temporal constraints. These constraints may cause conflict between services and affect the estimation over composition requirements. Existing verification approaches have not adequately addressed this issue. Therefore, this paper contributes to the verification method with temporal consistency checking and temporal satisfaction estimation. A set of checking rules and estimation formulae are presented according to workflow patterns and temporal dependencies. The method will lead to three major outcomes; consistent with satisfactory combination, consistent with unsatisfactory combination and inconsistent with unsatisfactory combination.

Keywords: composite service, temporal consistency, temporal constraints, temporal satisfaction, verification.

1 Introduction

A business process in the Web services environment contains a set of services that are composed using compositional language such as WS-BPEL[9]. The composed services may invoke external parties which offer required functionality based on certain quality criteria with some additional constraints.

This work focuses on the temporal constraints that can be attached together with the offered values. The temporal constraints can be utilized to restrict a period of time (e.g., execution duration, availability) or time points (e.g., start time operation, finish time operation, setup time operation) or both. In the Web services environment, these constraints can be extracted from Service Level Agreement(SLAs). Recently, a specification of SLA namely WS-Agreement [2] has been proposed and some recent works have used and extended it.

In our previous work [7], we have shown that temporal constraints may cause violation to individual temporal constraint of services when they are analyzed collaboratively. Thus, it is vital to check the temporal consistency and ensure that only consistent combinations are considered for the satisfaction estimation.

To address these problems, we propose a verification method which takes several major inputs including temporal constraints to perform consistency checking and temporal satisfaction estimation. This method utilises workflow patterns [15] as the basis to explore the process model. A set of consistency checking rules are proposed based on the temporal relation [1] [16] and temporal constraints verification in workflow [13] [4]. The temporal estimation formulae are proposed based on the QoS aggregation of service composition [3] [8]. The method will lead to three distinct outcomes; consistent with satisfactory combination, consistent with unsatisfactory combination and inconsistent with unsatisfactory combination.

The rest of the paper is organised as follows. Section 2 describes the data required for enabling the checking and estimation procedure. Section 3 presents the verification method with a set of checking rules and formulae as well as summarising them as an algorithm. Section 4 remarks some related works. Finally, section 5 summarises this paper and discusses some future work.

2 Capturing Related Information

The temporal consistency checking and the temporal satisfaction estimation requires various data as inputs. Information related to the execution order can be extracted from a composition model described using one of the composition languages such as WS-BPEL. The information related to QoS, temporal and composition requirement can be extracted from Web service level agreement specification such as WS-Agreement. Formally, the verification model involves a set of elements (P, F, T, U, G) where each of them is described as follows:

– P is a process represented as a graph with two elements (A, E). A is a set of activities $(s_1, ..., s_n)$ where each s_i refers to an individual service that provides one or more operations. E is a set of edges that define the relations between services, $E \subseteq A$ x A. In WS-BPEL, services can be identified through $< partnerlink >$. Furthermore, their relations can be determined by the control structure activities such as $< sequence >$, $< flow >$, etc.
– F is a function that retrieves a pattern as a subset of the process, S. For the sake of simplicity, this paper focuses on two types of patterns;
  - Sequential pattern which is comparable to $< sequence >$ in WS-BPEL. When the pattern type is sequential, S contains a pair of services, such as $(s_i \prec s_j)$.
  - Parallel pattern which is comparable to $< flow >$ in WS-BPEL. When the pattern type is parallel, S contains a partial set of services, such as $(s_i \prec (s_{i+1}, ..., s_{i+m}) \prec s_k)$.
– T is a function that retrieves temporal constraints information of a component service $s_i \in A$. We focus on two types of temporal constraints as follows:
  - start-after-time constraint sa; it constrains the time for a service $T_{sa}(s_i)$ to start executing.
  - finish-before-time constraint fb; it constrains the time for a service $T_{fb}(s_i)$ to finish executing.

  These constraints can be referred to the time period defined in WS-Agreement specification. Recent works such as [14] has discussed and proposed some extension to WS-Agreement. For instance, an agreement can be defined by two dates; start date and finish date. A start date does not mean an operation must start on the date, but rather states it can start at any time from this date. Similar interpretation can be given to the finish date. Obviously, these two dates can be translated into sa as the start date and fb as the finish date.

In addition to the agreement period, an agreement may also have a validity period for specific quality attribute. This data can be extracted through $< QualifyingCondition >$. For instance, a duration time of 2 hours can only be executed within office hour, 9am to 5pm. In regards to the temporal constraints, we may translate 9am as sa and 5pm as fb.

– U is a function that retrieves duration value of a component service $s_i \in A$, given as $U_d(s_i)$. A duration represents an interval value such as 10 minutes, 1 hour, etc. The value can be obtained from $< ServiceLevelObjective >$.
– G is a set of composition requirements. Since we are interested in time, the requirement would be any time-related requirement. In this work, we focus on the total time to be spent by the composed services.

3 Verification Approach

The following subsections provide the details of the temporal consistency checking and temporal satisfaction estimation which lead to the outcomes.

3.1 Temporal Consistency Checking

This procedure contains a set of rules which are based on the following:

– *Temporal relation* - As mentioned in [11] there are thirteen types of temporal relations formalised by [1] as interval relations and followed by [16] as point relations. For the sake of simplicity, we focus only on *before* and *overlap* temporal relations. Furthermore, we analyse the relation between temporal constraints based on point relations.
– *Estimation of Completion Time* - The estimated completion time will be computed and checked against fb. We utilize the works from temporal verification of workflow such as [13] [4]. However, in our work, the temporal constraint verification is meant to support determination of consistent combination between services.

With these rules, two distinct decisions can be defined, as follows:

Consistent Combination, CON -
It is determined when any of the following rule is satisfied, given as follows:

1. R_1: The fb of preceding service maintains the *before* type relation with the sa of succeeding service.
2. R_2: The addition of estimated completion time of the preceding service and the duration of succeeding service does not violate the fb of succeeding service.

These rules are applied to check the consistency of sequential, splitting and merging patterns. For the sequential pattern with a pair of services $(s_i \prec s_j)$, consistent combination, given as CON_{seq}, is determined when it satisfies the following:

$$R_1(T_{fb}(s_i) < T_{sa}(s_j)) \vee R_2(T_{sa}(s_i) + U_d(s_i) + U_d(s_j) <= T_{fb}(s_j)) \quad (1)$$

For the splitting pattern with a set of services $(s_i \prec (s_{i+1}, ..., s_{i+m}))$, consistent combination, given as CON_{split}, is determined when it satisfies the following, $\forall\ j = \{1, ..., m\}$:

$$R_1(T_{fb}(s_i) < T_{sa}(s_j)) \vee R_2(T_{sa}(s_i) + U_d(s_i) + U_d(s_j) <= T_{fb}(s_j)) \quad (2)$$

For the merging pattern with a set of services $((s_{i+1}, ..., s_{i+m}) \prec s_k)$, consistent combination, given as CON_{merge}, is determined when it satisfies the following, $\forall\ j = \{1, ..., m\}$:

$$R_1(T_{fb}(s_j) < T_{sa}(s_k)) \vee R_2(T_{sa}(s_j) + U_d(s_j) + U_d(s_k) <= T_{fb}(s_k)) \quad (3)$$

Combination of splitting and merging determine the consistency of the parallel pattern. Therefore, for the parallel pattern, given as CON_{par}, consistent combination is determined when it satisfies this rule:

$$CON_{split} \wedge CON_{merge} \quad (4)$$

Inconsistent Combination, INC -

It is determined when the combination violates R_1 AND R_2 for any pattern. The rules can be summarized as follows:

$$\neg CON_{seq} \vee \neg CON_{split} \vee \neg CON_{merge} \quad (5)$$

3.2 Temporal Satisfaction Estimation

This procedure will eventually lead to the overall conclusion namely; (i) consistent with satisfactory combination, (ii) consistent with unsatisfactory combination and (iii) inconsistent with unsatisfactory combination. We adopt the work from QoS aggregation of service composition [3] [8] to design the formulae. With the given formulae, the estimated time to be spent can be accumulated based on QoS workflow reduction techniques such as the one proposed in [3]. However, as discussed in our previous work [7], the temporal constraints may cause some waiting time. In this work, we define it as follows:

Definite Waiting Time, dwt -

It is determined when the fb of preceding service maintains the *before* type relation with the sa of succeeding service. Therefore, for the sequential pattern, the definite waiting time is computed as follows:

$$dwt_{seq} = T_{sa}(s_j) - T_{fb}(s_i) > 0 \quad (6)$$

For the splitting pattern, the definite waiting time is computed as follows:

$$dwt_{split} = \min(T_{sa}(s_{i+1}), ..., T_{sa}(s_{i+m})) - T_{fb}(s_i) > 0 \tag{7}$$

For the merging pattern, the definite waiting time is computed as follows:

$$dwt_{merge} = T_{sa}(s_k) - \max(T_{fb}(s_{i+1}), ..., T_{fb}(s_{i+m})) > 0 \tag{8}$$

For the parallel pattern, the definite waiting time is computed as follows:

$$dwt_{par} = dwt_{split} + dwt_{merge} \tag{9}$$

With this definition and formulae, the temporal satisfaction estimation for the total time to be spent can be obtained as follows:

Estimated Time Spent, $EstS$ -
For the sequential pattern, the estimated time spent is formulated as follows:

$$EstS_{seq} = U_d(s_i) + U_d(s_j) + dwt \tag{10}$$

For the parallel pattern, the estimated time spent is formulated as follows:

$$EstS_{par} = U_d(s_i) + \max(U_d(s_{i+1}), ..., U_d(s_{i+m})) + U_d(s_k) + dwt \tag{11}$$

By applying the workflow reduction technique, the estimated time to be spent can be accumulated. Therefore, the overall conclusion can be defined as follows:

Consistent with Satisfactory Combination, $CON_{satisfy}$ -
Given a process P, consistent with satisfactory combination is determined as follows:

$$for\ any\ patterns,\ (CON_{seq} \wedge CON_{par}) \wedge (EstS_{seq} + EstS_{par} \leq G) \tag{12}$$

Consistent with Unsatisfactory Combination, $CON_{unsatisfactory}$ -
Given a process P, consistent with unsatisfactory combination is determined as follows:

$$for\ any\ patterns,\ (CON_{seq} \wedge CON_{par}) \wedge (EstS_{seq} + EstS_{par} > G) \tag{13}$$

Inconsistent with Unsatisfactory Combination, $INC_{unsatisfactory}$ -
Given a process P, inconsistent with unsatisfactory combination is determined as follows:

$$for\ any\ patterns,\ (INC_{seq} \vee INC_{par}) \vee (EstS_{seq} + EstS_{par} > G) \tag{14}$$

3.3 The Algorithm

This section presents the algorithm for verifying composite services with temporal consistency checking and temporal satisfaction estimation as depicted in (1). The algorithm begins by extracting a pattern from P (line 1). It is followed by an iterative verification based on sequential or parallel pattern (lines 3-9 and 10-17). For both, the verification performs the temporal consistency checking (lines 4 and 11). If consistent, then it moves to the estimation (lines 5 and 12). Consistent services are then combined towards atomic service (lines 5 and 12-13). For the successful iteration, the algorithm checks whether the process can satisfy the composition requirement which leads to consistent with satisfactory or consistent with unsatisfactory (lines 22 and 24). For the unsuccessful iteration, the outcome is inconsistent with unsatisfactory combination (line 27).

```
**Require:** $(P, F, T, U, G)$
**Ensure:** $(CSC, CUC, IUC)$
 1: $F_{type}(S) \leftarrow$ get the first pattern in $P$
 2: **while** $P$ is not an atomic service **do**
 3:    **if** the pattern is $F_{seq}(s_i \prec s_j)$ **then**
 4:       **if** consistency checking is $CON_{seq}$ **then**
 5:          $U_d(s_i) \leftarrow EstS_{seq}$ and $s_i = s_i + s_j$ and Update $P$ with new $s_i$
 6:       **else**
 7:          Mark as inconsistent and break
 8:       **end if**
 9:    **end if**
10:    **if** the pattern is $F_{par}(s_i \prec (s_{i+1}, ..., s_{i+m}) \prec s_k)$ **then**
11:       **if** consistency checking is $CON_{par}$ **then**
12:          $U_d(s_i) \leftarrow EstS_{par}$ and $s_i = s_i + (s_{i+1}, ..., s_{i+m}) + s_k$ and
13:          Update $P$ with new $s_i$
14:       **else**
15:          Mark as inconsistent and break
16:       **end if**
17:    **end if**
18:    $F_{type}(S) \leftarrow$ get the next pattern in $P$
19: **end while**
20: **if** $P$ is an atomic service **then**
21:    **if** satisfaction checking is $CON_{satisfy}$ **then**
22:       **return** $CSC$
23:    **else**
24:       **return** $CUC$
25:    **end if**
26: **else**
27:    **return** $IUC$
28: **end if**
```

Algorithm 1. Verification Algorithm

4 Related Works

Verification of composite services is gaining attention from researchers. The taxonomy proposed in [5] is utilised for presenting this section. In relation to our work, two perspectives are involved, namely the performance perspective and temporal perspective.

From the performance perspective, the aim is typically to predict the satisfaction of QoS values over the performance metric. In the context of web service composition, existing work such as [3] [8] proposed measurement methods based on workflow structures such as sequential, parallel, etc. Our work adopts this direction by emphasizing the needs to check for temporal constraint consistency which may give impact to the prediction. Moreover, it is worth to highlight recent work that addressed the integration of composition specification and QoS-related specification to support performance prediction. In [6], an extension of WSDL was proposed to describe and integrate QoS information into BPEL processes. In addition, [17] addressed the integration of information gathered from BPEL and SLA and used the simulation technique to verify against performance metric.

From the temporal perspective, the aim is to ensure the given temporal constraints are consistent when combining with the composition specification. Within the web service composition, [10] addressed verification of BPEL process with temporal constraints which was presented as finite-state machine equipped with a set of clock variables. Then, the verification of temporal constraints was implemented by using a model checker. In addition, [12] proposed temporal constraints that were defined in OWL-S by using time ontology. Then, an extension of Petri Nets was proposed and used to verify the temporal constraints. In contrast to [10] [12], we consider different types of temporal constraints that can be extracted from SLAs. Furthermore, we utilised workflow-based pattern instead of formal method.

5 Conclusion and Future Work

In this paper, we have presented a verification method that utilises various information including temporal constraints and structural dependencies to perform the checking and estimation procedure. The method will lead to three major outcomes. In addition, it can contribute to the analysis of the behavior of composite services from the temporal consistency and satisfaction perspective. Obviously, there is no point of making a prediction if services are actually inconsistent between each other.

Future work will focus on the evaluation of the effectiveness of the approach. Furthermore, we will explore other properties such as related to resource that contribute to the consistency checking and satisfaction estimation. This will expand the method to cater from different perspectives.

Acknowledgments. This research is partly supported by the Australian Research Council Discovery Project Scheme under grant number DP0663841 and University of Wollongong Small Research Grant.

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM 26(11), 832–843 (1983)
2. Andrieux, A., Czajkowski, K., Dan, A., et al.: Web services agreement specification (ws-agreement). In: Open Grid Forum (2007)
3. Cardoso, J., Sheth, A., Miller, J., et al.: Quality of service for workflows and Web service processes. Web Semantics: Science, Services and Agents on the World Wide Web 1(3), 281–308 (2004)
4. Chen, J., Yang, Y.: Temporal dependency for dynamic verification of temporal constraints in workflow systems. In: Jin, H., Pan, Y., Xiao, N., Sun, J. (eds.) GCC 2004. LNCS, vol. 3251, pp. 1005–1008. Springer, Heidelberg (2004)
5. Chen, J., Yang, Y.: A taxonomy of grid workflow verification and validation. Concurrency and Computation: Practice and Experience 20(4), 347–360 (2008)
6. D'Ambrogio, A., Bocciarelli, P.: A model-driven approach to describe and predict the performance of composite services. In: Proc. WOSP 2007, pp. 78–89 (2007)
7. Ismail, A., Yan, J., Shen, J.: Dynamic service selection for service composition with time constraints. In: Proc. ASWEC 2009, pp. 183–190 (2009)
8. Jaeger, M.C., Rojec-Goldmann, G., Muhl, G.: Qos aggregation for Web service composition using workflow patterns. In: Proc. EDOC 2004, pp. 149–159 (2004)
9. Jordan, D., Evdemon, J., Alves, A., et al.: Web services business process execution language version 2.0. OASIS (2007)
10. Kazhamiakin, R., Pandya, P., Pistore, M.: Representation, Verification, and Computation of Timed Properties in Web Service Compositions. In: Proc. ICWS 2006, pp. 497–504 (2006)
11. Koubarakis, M.: Temporal CSPs. In: Handbook of Constraint Programming, pp. 665–697. Elsevier, Amsterdam (2006)
12. Liu, R., Dai, G., Hu, C., et al.: A Verification Method for Temporal Consistency of Service Flow. In: Proc. COMPSAC 2008, pp. 1187–1192 (2008)
13. Marjanovic, O.: Dynamic verification of temporal constraints in production workflows. In: Proc. ADC 2000, pp. 74–81 (2000)
14. Müller, C., Martín-Díaz, O., Ruiz-Cortés, A., Resinas, M., Fernández, P.: Improving temporal-awareness of WS-agreement. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 193–206. Springer, Heidelberg (2008)
15. van der Aalst, W.M.P., Hofstede, A.H.M.T., Kiepuszewski, B., et al.: Workflow patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
16. Vilain, M., Kautz, H.: Constraint propagation algorithms. In: Proc. AAAI 1986, pp. 377–382 (1986)
17. Xiao, H., Chan, B., Zou, Y., et al.: A Framework for Verifying SLA Compliance in Composed Services. In: Proc. ICWS 2008, pp. 457–464 (2008)