Deriving Transactional Properties of Composite Web Services

Li Li and Chengfei Liu
Centre for Information Technology Research
Swinburne University of Technology
Melbourne, VIC 3122, Australia
{lli, cliu}@ict.swin.edu.au

Junhu Wang
School of ICT
Griffith University
Gold Coast, QLD 4215, Australia
j.wang@griffith.edu.au

Abstract

Web services have been emerging as a promising technology for business integration. Transactional support to integrated businesses via composing individual Web services is a critical issue. Current Web services protocols (e.g. BPEL4WS) have been proposed to deal with this issue on a strong assumption that each Web service is compensatable for a recovery purpose. It is arguable that Web services composition requires more transactional support beyond the compensation-based solution. This paper looks into the problem of transactional support for composing and scheduling those Web services that may have different transactional properties. The transactional properties of workflow constructs, which are fundamental to the composition of Web services, are thoroughly investigated. The concept of a connection point is introduced to derive the transactional properties of composite Web services. The scheduling issue of composite Web services is also discussed.

1 Introduction

Firms are constantly looking for ways to survive and thrive in today's e-business. Nowadays enterprises are willing to outsource their internal business processes as services and make them accessible via the Web. By doing so, they can dynamically combine individual services to provide new value-added services. In order to achieve this, transactional support has become more important for Web service technology. Moreover, flexible transaction strategies must be enforced to allow consistent and reliable business process executions.

Most business-to-business applications require transactional support in order to guarantee consistent outcome and correct execution. These applications often involve long-running computations, loosely-coupled systems, and components that do not share data, location, or administration. It is therefore difficult to incorporate atomic transactions (ACID) [17] within such architectures.

Web services have been emerging as a promising technology for business process integration. Transactional support to business integration via composing individual Web services is a critical issue. Several transaction relevant Web service standards have been proposed. Among them are BPEL4WS [1], WS-Coordination [5], WS-Transaction (including WS-AT and WS-BA) [6, 7, 8], and WSCI [2]. These Web services protocols have been proposed to deal with this issue on a strong assumption that each Web service is compensatable for a recovery purpose. In other words, compensation is the basic mechanism adopted by all of these standards for backward recovery. However, it is arguable that Web services composition requires more transactional support beyond the compensation-based solution.

With long-running systems where exclusively locking resources is impossible or impractical, investigating transactional properties of composed Web services is becoming more important with the availability of different Web services and the applicable standards. Intuitively, Web services infrastructures should provide comprehensive transaction support so that Web services can compose other Web services in a transactional manner. However, the fact is that the current Web services efforts only provide limited support without giving much thought to the transactional features [23]. It is time to analyse the transactional behaviours of composite services with the presence of Web services and their transactional properties. Because transaction requirements highly reflect the underlying business specifications which are demonstrated by composite Web services in a business process, it is also necessary to look into workflow constructs such as sequential, concurrent, alternative and iterative routings in terms of composition.

To this end, transactional features of workflow constructs will be studied followed by the discussion of composing Web services via these constructs in a Web service environment. Ultimately, transactional scheduling issues will be targeted to guarantee consistent outcomes and reliable business processes. As such, this paper makes main contributions in deriving transactional properties and scheduling transactional business process.

The rest of the paper is organised as follows. Section 2 is a literature review with the focus on the current Web service transaction proposals. Section 3 introduces the technical background to be used in this paper. Section 4 analyses the transactional properties of workflow constructs. Section 5 further explores the transactional properties of composite Web services. Particular attention has been given to the transactional support for composing and scheduling Web services. Section 6 concludes this paper.

2 Related Work

Transactional support for business integration via composing of individual Web service is a critical issue. Current Web services specifications, for example BPEL4WS, have been proposed to deal with this issue but on a strong assumption that each Web service is compensatable for a recovery purpose.

The mechanism of compensation is originally proposed by Gray [11], and then widely used in both advanced transaction models (ATMs) [9, 10] and transactional workflows [13] to maintain atomicity when the isolation property has to be relaxed.

Both the WS-Coordination and WS-Transaction (WS-T) specifications complement BPEL4WS to provide mechanisms for defining specific standard protocol to transaction processing systems or other applications. They target the coordination of multiple Web services. In addition, they provide a Web service-based approach to improve the dependability of automated long-running business transactions in an extensible and interoperable way. WS-Coordination provides a framework for coordinating the actions of distributed applications via coordination-context sharing. WS-Transaction leverages WS-Coordination by describing the coordination types that are used with the extensible coordination framework outlined in WS-Coordination specifications. There exist two coordination types, namely AtomicTransaction (AT) and BusinessActivity (BA). WS-AT, which provides protocol for short-lived atomic units of work but not suitable for Web service transactions.

It is important to note that the BA model derives from a specific industry requirement in the BPEL4WS specification. For example, WS-BA's compensation model is based on open nested transaction. Hence, WS-T BA lacks equity between services as a parent scope has the ability to select which child tasks are to be included for a specific business activity. In doing so, parents direct the orchestration and impose their transactional semantics to children services. Furthermore, it seems that ATs and BAs may be sufficient for the current use cases that the specifications are aimed at. It is generally accepted that other protocols may well be needed later.

Other efforts which aim at addressing business transactions in a loosely coupled Web service environment include the OASIS Business Transaction Protocol (BHP) [19] and the W3C Tentative Hold Protocol (THP). However, BHP attempts to solve a variety of different problems by only one model (atoms can be regarded as special case of cohesions, the cohesion model is essentially a superset of the atom model). This has the disadvantage of not allowing reasoning about an applications's overall functionality and behaviour. It also has a potential problem of transaction interoperability between services implemented with different approaches. THP, on the other hand, is likely to minimise compensation, but there is no guarantee on holding a resource and it must work with other protocols in practice. Efforts from ETH [21] also deal with the problems of concurrent control and recovery for transactional processes. However, no attempt has been made to handle these problems in a Web service environment.

It is highly recommended that some mechanisms to be proposed to relax ACID properties. The compensation-based solution[12] is well-known in sematically eliminating the effects of all the operations it has performed. Bhiri et al [3] proposed an approach by using accepted termination states (ATS) property as a correctness criterion to relax atomicity. Work reported in [4] proposed a transactional approach for reliable Web services compositions by ensuring the failure atomicity required by the designers. A set of transactional rules have been defined to assist designers to compose a valid composite Web services with regards to the specified ATS.

A reservation-based extended transaction protocol is presented in [24] by Zhao et al. The proposed protocol avoids the use of compensating transactions but defines every task within a business activity to be executed in two steps as traditional short-running transactions. Our approach, however, is centred on deriving transactional properties of composite Web services to achieve the schedulability. Eventually, a desirable scheduling regarding the transactional properties can be achieved.

In terms of workflow control flow patterns, the paper [22] presents well-supported patterns which can be borrowed to describe different types of Web service composition.

3 Preliminaries

A Web service is a self-contained modular program that can be discovered and invoked across the Internet. Web services provide a means for different organisations to connect their applications with one another to conduct business spanning organisation boundaries regardless of platforms and languages.

The current Web services specifications define protocols for Web services interoperability. Web services frequently tie together large number of participants to form distributed applications. Therefore, the resulting business applications will end with complex structure and relationships between participants. Additionally, the overall behaviours will be more complicated than that of a single Web service. New value-added services are possible by orchestrating individual Web services. It is known that business processes can be implemented by workflow technologies. Generally, a business process is composed of smaller workflow fragments in the form of $wf_1, wf_2, \ldots, wf_n$. In a Web services environment, a workflow or a workflow fragment can be represented as a composite Web service, and an activity of a workflow can be implemented by invoking a Web service. These terms may be used interchangeably in the following discussion.

An activity in a workflow may or may not be transactional [15, 16]. Activities are either atomic or composite. A composite activity is composed of a set of Web services of which it may be atomic or composite. In this perspective, we discuss basic workflow constructs in deriving transaction properties in Web services environments.

Transactions are one of the most fundamental concepts in delivering reliable application processing. However, the transactions in business processes are different from traditional transactions which support ACID properties. Characteristics such as long-running, business latencies, and network latencies may prevent Web services transactions from abiding to the strict ACID properties. In a long-running system, the transactional properties of an activity may reveal the following three features [18]: **compensatable** (c), **retriable** (r), and **pivot** (p). More precisely, we have the following definitions.

Definition 1 **(Compensatable)**. An activity is compensatable if it is able to offer compensation policies to semantically undo the original activity.

Definition 2 **(Retriable)**. An activity is retriable if it is able to offer forward recovery. In other words, activities with this property can guarantee a successfully termination after a finite number of invocations.

Definition 3 **(Pivot)**. An activity is pivot if it is neither compensatable nor retriable. On one hand, there is no guarantee that this type of activity can be executed successfully. On the other hand, a committed pivot activity cannot be rolled back.

In this paper, a composite transactional Web service means a conglomeration of transactional Web services to offer new value-added services to business applications.

Below are further descriptions of composite Web services in terms of workflow constructs.

- **Sequential Composite Services**: The activities within a sequential composite service are executed with a sequential dependency between them. As a result, the transactional property of an activity will be affected by those introduced previously (i.e. the properties of the preceding activities), especially the immediate preceding activities.
- **Concurrent Composite Services**: Different paths of concurrent composite services are allowed to execute simultaneously. Since there is no dependent restrictions between these paths, it leaves the room for the system to schedule these activities independently.
- **Alternative Composite Services**: One and only one path of alternative composite services will be executed.
- **Iterative Composite Services**: The activities within an iterative composite service are executed with repetition.

4 Transactional Properties of Workflow Constructs

We are interested in properly scheduling Web services in a transactional manner. The scheduler exploits a combination of transactional Web services by treating the execution of them as an unit of work. We say that a composite Web service is schedulable if it can be treated as an unit of work. In our discussion, particular concerns are given to the scheduling of transactional workflow with the presence of transactional properties in $\{c, r, p\}$ for each individual Web services. In the following, $\overrightarrow{s}$ is used to indicate that the transactional property of a workflow is schedulable whilst $\widetilde{s}$ is non-schedulable. In terms of the transactional properties in $\{c, r, \overrightarrow{s}\}$, we believe properties c and r are as much desirable than $\overrightarrow{s}$.

The transactional properties of a composite Web service can be derived from those of individual services given that they conform to $\{c, r, p\}$. Regarding the transactional properties, an activity's transactional property p neither supports forward nor backward recovery regarding scheduling, whilst r is able to be forward and c to be backward. In the following, we omit the activity itself but only illustrate its transactional property for simplicity. In other words, the notation $c \to r$ stands for two activities in a sequential routing, with transactional properties c and r, respectively.

A key consideration should be put on workflow constructs and how various workflow fragments are linked to form a workflow. From the literature review, it is clear that the vast majority of workflow languages support the sequence, splits (AND and OR), joins (AND and OR) and iteration [14, 20, 22]. As our focus is on the description of a workflow as an execution from the start point to the end point, all these constructs will be discussed below.

Table 1. Transaction property of sequential routing

t_1	t_2	t
p	p	$\widetilde{s}$
p	r	$\overrightarrow{s}$
p	c	$\widetilde{s}$
r	p	$\widetilde{s}$
r	c	$\widetilde{s}$
c	p	$\overrightarrow{s}$
c	r	$\overrightarrow{s}$
c	c	c
r	r	r

Since different transactional properties combinations may be presented given there are different workflow constructs, it is worth looking into them one by one.

(1) **Sequential routing**

A symbol $t_1 \rightarrow t_2$ $(t_1, t_2 \in \{c, r, p\})$ is defined to represent a sequential routing of two activities a_1 followed by a_2, and with transactional properties t_1 and t_2, respectively. The derived transactional property t of this sequential routing is shown in Table 1.

It is worth noting that sometimes having n times repetition of same transactional property will lead to the same transactional property as a whole. Take the sequential routing in a workflow for example. Apparently, $c \rightarrow c \rightarrow c \ldots \rightarrow c$ will result in c which is schedulable without a doubt. The same is true for other two routings (i.e. concurrent and alternative routings). Similarly, the above discussions also apply to the combination of $r \rightarrow r \rightarrow r \ldots \rightarrow r$ which leads to r at the end. As such, activities with the same transactional properties (i.e. $\{c, r\}$) are semantically equivalent as to a single activity in terms of their transactional properties. Without loss of generality, these activities are eligible to be grouped and represented as c and r respectively.

(2) **Concurrent routing**

A symbol $\binom{t_1}{t_2}$ $(t_1, t_2 \in \{c, r, p\})$ is defined to represent a concurrent routing of two activities a_1 and a_2 to be executed simultaneously. Activities a_1 and a_2 have transactional properties t_1 and t_2 , respectively. The derived transactional property t of this concurrent routing is shown in Table 2.

A concurrent routing is defined such that the involved activities can be executed independently (Section 3). The *scheduler* has freedom to determine which path to go first in order to achieve a better transactional property. The following are examples.

Table 2. Transaction property of concurrent routing

t_1	t_2	t
p	p	$\widetilde{s}$
c	c	c
r	r	r
p	r	$\overrightarrow{s}$
r	p	
p	c	$\overrightarrow{s}$
c	p	
r	c	$\overrightarrow{s}$
c	r	

Table 3. Transaction property of alternative routing

t_1	t_2	t
p	p	p
p	r	r
p	c	c
r	c	$\{r, c\}$

- *Case 1*: A concurrent routing of $\binom{p}{r}$ or $\binom{r}{p}$. The schedulable feature $\overrightarrow{s}$ cannot be achieved unless an activity with transactional property p is be executed before an activity with r.
- *Case 2*: A concurrent routing of $\binom{p}{c}$ or $\binom{c}{p}$. The schedulable feature $\overrightarrow{s}$ can be achieved only if an activity with transactional property c to be executed before an activity with p.
- *Case 3*: A concurrent routing of $\binom{c}{r}$ or $\binom{r}{c}$. The schedulable feature $\overrightarrow{s}$ can only be achieved only if an activity with transactional property c to be executed before an activity with r.
- *Case 4*: A concurrent routing of $\binom{r}{r}$ and $\binom{c}{c}$. The schedulable feature $\overrightarrow{s}$ can be achieved with r and c, respectively.
- *Case 5*: A concurrent routing of $\binom{p}{p}$. The schedulable feature $\overrightarrow{s}$ cannot be achieved at all.

Cases 1, 2 and 3 demonstrated that a concurrent routing is schedulable under certain circumstances. It is called *conditionally* schedulable. On the contrary, scheduling in case 4 is called *unconditionally* schedulable.

(3) **Alternative routing**

A symbol $\left\langle{t_1 \atop t_2}\right\rangle$ $(t_1, t_2 \in \{c, r, p\})$ is defined to represent an alternative routing of two activities a_1 and

a_2 with transactional properties t_1 and t_2, respectively. The derived transactional property t of this alternative routing is shown in Table 3.

Since properties such as c, r, p are treated more desirable and sometimes more meaningful than just $\overrightarrow{s}$, the chance to have any transactional property from $\{c, r, p\}$ is preferable in scheduling. For example, in a case where the transactional properties of two paths are all p, the better outcome can be achieved if it is followed by a service with transactional property r (see Table 1). Therefore, it is wise to delay deciding about the transactional property until considering connecting transaction properties. On the other hand, we would think that a single property p is better than $\overrightarrow{s}$, which is obtained by scheduling properties c, p and r properly. In this sense, it is our belief that an order $\{c, r\} \succ p \succ \overrightarrow{s} \succ \widetilde{s}$ holds.

(4) **Iterative routing**

A symbol $(t_1)^*$ ($t_1 \in \{c, r, p\}$) is defined to represent an iterative routing of activity a_1 with a transactional property t_1. The derived transactional property can be treated as those of a sequential routing with repetition.

In our discussion, the deriving transactional property of an iterative routing is a special case of a sequential routing. Details will be given in the following section.

5 Transactional Properties of Composite Services

Since the transactional property of a workflow highly depends on the integration of various workflow fragments, it is interesting to extend transactional properties discussed above to include more complicated fragments.

With this in mind, we assume that different workflow fragments are available. In addition, each of these workflow fragments has a transactional property which can be deduced from Tables 1, 2 and 3. In the following, we are going to answer what an overall transactional property of a workflow would look like if each workflow fragment is schedulable.

We can compose sequential routing of fragments wf_1 and wf_2 to construct a workflow. A composed sequential routing is represented in the form of $wf_1 \rightarrow wf_2$. A composed concurrent routing is represented in the form of $\binom{wf_1}{wf_2}$, whilst a composed alternative routing is in the form of $\left\langle {wf_1 \atop wf_2} \right\rangle$. Similarly, a composed iterative routing is represented as $(wf_1)^*$.

5.1 Composing Sequential Routing

Given that two workflow fragments wf_1 and wf_2 with each of them is schedulable (Figures 1(a), 2(a)), the following discussion aims to determine whether the composed Web services (Figures 1(b), 2(b)) are schedulable or not. To make it clear, let us look at two examples shown below.

- *Example 1*: wf_1 is in the form of $(\ldots \rightarrow p)$, while wf_2 is with $(p \rightarrow \ldots)$. According to Table 1, it is apparent that $(\ldots \rightarrow p)$ and $(p \rightarrow \ldots)$ which leads to the composed Web services non-schedulable because $p \rightarrow p$ is non-schedulable (Table 1).

- *Example 2*: wf_1 is in the form of $(\ldots \rightarrow r)$, while wf_2 is with $(c \rightarrow \ldots)$. The composed Web services is non-schedulable because $r \rightarrow c$ is non-schedulable (Table 1).

Therefore, the fact that all workflow fragments are schedulable is insufficient in solving the scheduling of the composition of these fragments. Some efforts are needed to make the schedulable feature decidable. It is crucial to examine some important points, for instance, those within the dotted circles as shown in Figures1(b) and 2(b).

Definition 4 **(Connection Point)**. Given that a sequential composition of two fragments wf_1 and wf_2 that are schedulable, a *connection point* is defined as a point to connect these two fragments in a workflow.

Within a workflow fragment, a *connection point* has the same meanings but connecting two activities rather than two workflow fragments.

Definition 5 **(Scheduling Bridge)**. Given that a sequential composition of two fragments wf_1 and wf_2 that are schedulable, a *scheduling bridge* consists of the last activity of wf_1, the *connection point* between wf_1 and wf_2, and the first activity of wf_2.

Definition 6 **(Safe Connection Point)**. Given that a sequential composition of two fragments wf_1 and wf_2 that are schedulable, a *safe connection point* is a *connection point* such that the *scheduling bridge* is schedulable.

Lemma 1: It is clear that any *connection point* within a schedulable workflow fragment is a safe *connection point.*

Theorem 1: Given that two workflow fragments wf_1 and wf_2 that are schedulable, the sequential composition of wf_1 and wf_2 (i.e. $wf_1 \rightarrow wf_2$) is schedulable iff a *connection point* between wf_1 and wf_2 is a safe *connection point.*

Proof of Theorem 1. Let us first look into the proof of "$\rightarrow$".

Note that the composed service is schedulable. According to *Lemma 1*, the *connection point* between wf_1 and wf_2 is hence a safe *connection point.*

Coming next is the proof of "$\leftarrow$".

Note that the *connection point* is safe. According to the definition of *safe connection point*, the *scheduling bridge* between wf_1 and wf_2 is schedulable. In order to prove that the composition of wf_1 and wf_2 is schedulable, all activities from both wf_1 and wf_2 should be considered.

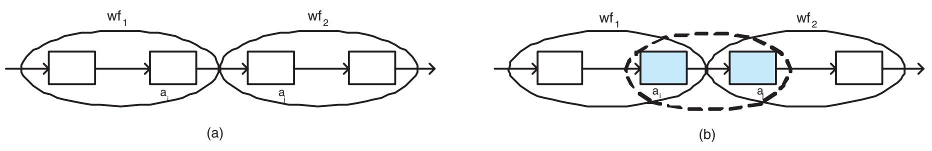

Figure 1. Sequential composition - case 1

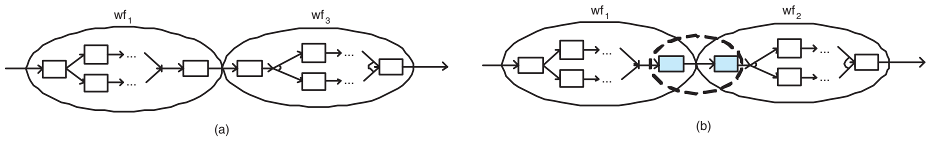

Figure 2. Sequential composition - case 2

(1) Backward recovery: It is evident that the backward recovery is able to continue because any *connection point* within wf_1 is a safe *connection point*.

(2) Forward recovery: Since wf_2 is schedulable, all *connection point* should be safe according to *Lemma 1*, it is certain that the forward procedure can be guaranteed to the end point.

As a result, the sequential composition of wf_1 and wf_2 is schedulable. □

Theorem 1 is not limited to two workflow fragments. In the sequential composition with a series of workflow fragments, for example, $wf_1 \rightarrow wf_2 \rightarrow wf_3$, the safe *connection point* concept applies to determine whether the composition of these workflow fragments is schedulable or not. From the above discussion, we could see that a *connection point* has a significant influence on Web services composition scheduling.

5.2 Composing Concurrent Routing

Assume that wf_1, wf_2, and wf_3 are schedulable. Now let us consider a composition of these workflow fragments with a concurrent routing (Figure 3).

Theorem 2: Given that workflow fragments wf_1, wf_2, wf_3 are schedulable. If the composition of a concurrent routing of wf_1 and wf_2 (i.e. $\binom{wf_1}{wf_2}$) is schedulable, then a new composition consisting of a concurrent routing of wf_1 and wf_2 and another fragment wf_3 in the form of $\binom{wf_1}{wf_2} \rightarrow wf_3$ is schedulable iff a *connection point* is safe for two *scheduling bridges* over $(wf_1 \rightarrow wf_3)$ and $(wf_2 \rightarrow wf_3)$.

Proof of Theorem 2. The proof of "→" is clear.

Below is the proof of "←".

It is necessary to consider what would be happening in terms of the transactional property as a whole. It is known that a *connection point* is safe over the bridges of $(wf_1 \rightarrow wf_3)$ and $(wf_2 \rightarrow wf_3)$.

(1) Backward recovery: Let *connection point* between $(wf_1 \rightarrow wf_3)$ and $(wf_2 \rightarrow wf_3)$ be ck. Because ck is safe for both bridges, it allows the recovery to head for either wf_1 or wf_2 via the bridge. Take another *connection point* in wf_1 for example, the recovery mechanism works well because wf_1 is schedulable. The same is true for a connection point in wf_2.

Moreover, since $\binom{wf_1}{wf_2}$ is schedulable, ck is able to move back to any connection point within $\binom{wf_1}{wf_2}$. As each individual fragment is schedulable, the backward recovery will proceed smoothly.

(2) Forward recovery: Forward recovery is guaranteed because wf_3 is schedulable.

As a result, $\binom{wf_1}{wf_2} \rightarrow wf_3$ is schedulable. □

In a concurrent composition, the *scheduler* may choose an option to execute activities with c property first since it can guarantee the process semantically undo what has been performed previously if it failed. Therefore, a path with c transactional property has a preference over others. What is to be executed first is just a scheduling strategy without giving much thought to concurrency control. Still, the overall transaction property relies on all paths. This is the reason behind what we discussed above.

5.3 Composing Alternative Routing

In terms of composing an alternative routing with another fragment, if the composite activity is able to be executed from the start point to the end point via one of its paths, then the schedulable feature is guaranteed.

Theorem 3: Given that workflow fragments wf_1, wf_2, wf_3 are schedulable. If the composition of an alternative routing of wf_1 and wf_2 (i.e. $\left\langle {wf_1 \atop wf_2} \right\rangle$) is schedulable, then the new composition consisting of the alternative routing and wf_3 in the form of $\left\langle {wf_1 \atop wf_2} \right\rangle \rightarrow wf_3$ is schedulable iff the *connection point* is safe for at least one of *scheduling bridges* over $(wf_1 \rightarrow wf_3)$ and $(wf_2 \rightarrow wf_3)$.

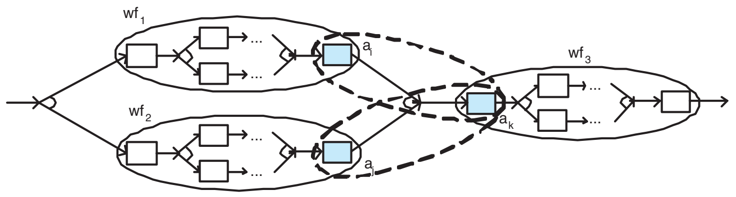

Figure 3. Concurrent composition

The proof is omitted (see the proof of *Theorem 2*).

5.4 Composing Iterative Routing

When composing iterative routing $(wf_1)^*$ with another fragment wf_2 in the form $(wf_1)^* \rightarrow wf_2$, the transactional property of the composed fragments is equal to that of a sequential composition in the form of $wf_1 \rightarrow wf_1 \rightarrow wf_2$ which we have discussed in Section 5.1.

5.5 General Workflow

Given that a series of workflow fragments $wf_1, \ldots, wf_n$ are schedulable, a general workflow is denoted as Θwf_i $(1 \leq i \leq n)$. Here, Θ stands for a sequence of applications of constructs that belong to $\{\rightarrow, (), \langle\rangle, ()^*\}$. Note that we use wf_{i+1} to indicate an immediate successor of wf_i, while wf_i is an immediate preceding of wf_{i+1}.
Lemma 2 (Decidable): A composition of a series of workflow fragments, i.e. Θwf_i is schedulable or not is decidable.

Proof of Lemma 2. Whether a *connection point* is safe or not plays an important role in determining the transactional property of a composed workflow. It can be illustrated as follows.

Now let a composed workflow wf be in the form of Θwf_i $(1 \leq i \leq n)$, $wf' = wf \backslash wf_n$. Thus we have $wf = wf'\theta_{n-1}wf_n,\ \theta_{n-1} \in \{\rightarrow, (), \langle\rangle, ()^*\}$. Because the safe *connection point* concept can be used to analyse the transactional property of composing wf' with wf_n, and the transactional property of wf_n is decidable, so that we could look into the transactional property of wf' as a replacement. Similarly, we may define that there is wf'' in the form of $wf'' = wf \backslash (wf_{n-1}\theta_{n-1}wf_n)$. This allow us to have a smaller workflow wf'' to check for. In this way, more and more workflow fragments can be separated from workflow wf until we reach fragment wf_1. Since the transactional property of wf_1 is decidable, it is clear that the transactional property of wf is decidable. □

It is clear that the transactional property of a composed workflow can be derived by applying *Theorems 1, 2* and 3 properly. A safe *connection point* seems like a safeguard which would enable the transactional properties of a composed workflow to be determined as early as possible. Ultimately, it provides transactional support for scheduling Web services that have different transactional properties.

6 Conclusions

The motivation behind the above discussion has been the desire to tackle transactional requirements in Web service environments. It is arguable that Web services composition requires more transactional support beyond the compensation-based solution. As a result, it is highly beneficial to investigate the deployment of transactional properties especially when Web service composition is concerned.

In this paper, we concentrated on deriving the transactional properties of composite Web services. At the same time, attention has also been given to scheduling issues. We attempt to answer whether a workflow is schedulable or not and how to determine its transactional property.

Consequently, transactional properties of a workflow has been carefully studied at workflow constructs and composite Web services levels. *connection point* concept was defined to resemble the critical connecting point in Web service composition. We have discussed the scheduling issue in a Web service environment where traditional transaction protocols are insufficient to cope with. One of the remarkable characteristics of our discussion is that the transactional property of a composite Web service is decidable.

Acknowledgments

This work is supported by the Australian Research Council Discovery Project under the grant number DP0557572.

References

[1] T. Andrews, et al. Business Process Execution Language for Web Service (BPEL4WS) 1.1, May 2003. *http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.*

[2] A. Arkin, et al. Web Service Choreography Interface 1.0, August 2002. *http://www.w3.org/TR/wsci/.*

[3] S. Bhiri, C. Godart, O. Perrin. Reliable Web Services Composition using a Transactional Approach. In Proc. of 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05), pp. 15-21, 2005.

[4] S. Bhiri, O. Perrin, C. Godart. Ensuring Required Failure Atomicity of Composite Web Services, the International World Wide Web Conference (WWW2005), pp. 138-147, Chiba, Japan, May, 2005.

[5] L. F. Cabrera et al. Web Services Coordination (WS-Coordination), August 2005. *http://specs.xmlsoap.org/ws/2004/10/wscoor/wscoor.pdf.*

[6] L. F. Cabrera et al. Web Services Business Activity Framework (WS-BusinessActivity) 1.0, August 2005. *http://ftpna2.bea.com/pub/downloads/webservices/WS-BusinessActivity.pdf.*

[7] L. F. Cabrera et al. Web Services Atomic Transaction (WS-AtomicTransaction) 1.0, August 2005. *http://ftpna2.bea.com/pub/downloads/webservices/WS-AtomicTransaction.pdf.*

[8] W. Cox, F. Cabrera, et al. Web Service Transaction (WS-Transaction), January 2004. *http://dev2dev.bea.com/pub/a/2004/01/ws-transaction.html.*

[9] A. Elmagarmid (Ed.). Database Transaction Models for Advanced Applications, Morgan Kaufmann, 1992.

[10] H. Garcia-Molina, K. Salem. Sagas, In Proceedings of the ACM Conference on Management of Data, pp. 249-259, 1987.

[11] J. Gray. The transaction concept: Virtues and Limitations, In Proc. of the International Conference on Very Large Data Bases, Cannes, France, pp. 144-154, 1981.

[12] J. Gray and A.Reuter. Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993.

[13] Paul W. P. J. Grefen. Transactional Workflows or Workflow Transactions? In Proc. of the 13th International Conference on Database and Expert Systems Applications, LNCS 2453, pp. 60-69, 2002.

[14] P. Lawrence (eds). Workflow Handbook, Workflow Management Coalition. John Wiley and Sons, New York, 1997.

[15] C. Liu, M. Orlowska, X. Liu, X. Zhou. Improving Backward Recovery in Workflow Systems, In: Proc. of the 7th International Conference on Database Systems for Advanced Applications (DASFAA'01), pp. 276-283, Hong Kong, SAR, China, April 2001.

[16] C. Liu, X. Lin, M. Orlowska, and X. Zhou. Confirmation: Increasing Resource Availability for Transactional Workflows. *Information Science*, Vol 153, pp. 37-53, 2003.

[17] N. Lynch, M.Merritt, W.Weihl and A. Fekete. Atomic Transactions, Morgan Kaufmann, 1993.

[18] S. Mehrotra, R. Rastogi, A. Silberschatz, and H. Korth. A Transaction Model for Multidatabase Systems. In Proc. of the 12th International Conference on Distributed Computing Systems (ICDCS92), IEEE Computer Society Press, Yokohama, Japan, pp. 56-63, 1992.

[19] OASIS Committee Specification Business Transaction Protocol 1.0, 2002. *http://www.oasis-open.org/home/index.php.*

[20] M. Rusinkiewicz, A. Sheth. Specification and Execution of Transactional Workflows. In Modern Database Systems: The Object Model, Interoperability, and Beyond., W. Kim Ed., ACM Press and Addison-Wesley, 1995.

[21] H. Schuldt, G. Alonso, C. Beeri, H. J. Schek. Atomicity and Isolation for Transactional Processes. ACM Transactions on Database Systems, 27(1), pp. 63-116, 2002.

[22] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns, Distributed and Parallel Databases, 14(1), pp. 5-51, 2003.

[23] Z. Yang and C. Liu. Implementing a Flexible Compensation Mechanism for Business Processes in Web Service Environment, In Proc. of IEEE International Conference on Web Services (ICWS 2006), pp. 753-760, Chicago, USA, September 2006.

[24] W. Zhao, L. E. Moser and P. M. Melliar-Smith, A Reservation-Based Coordination Protocol for Web Services, In: Proc. of the IEEE International Conference on Web Services (ICWS'05), pp. 49-56. Orlando, FL, July 2005.