Local Analysis of Atomicity Sphere for B2B Collaboration*

Chunyang Ye
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
cyye@cse.ust.hk

S.C. Cheung§
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
scc@cse.ust.hk

W.K. Chan♣
Department of Computer Science
City University of Hong Kong
Tat Chee Avenue
Hong Kong, China
wkchan@cs.cityu.edu.hk

Chang Xu
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
changxu@cse.ust.hk

ABSTRACT

Atomicity is a desirable property for business processes to conduct transactions in Business-to-Business (B2B) collaboration. Although it is possible to reason about atomicity of B2B collaboration using the public views, yet such reasoning requires the presence of a trustworthy party who has complete knowledge of these views. It is inapplicable when some parties may want to keep the confidentiality of their collaborative partners for privacy and other business reasons, or the trustworthy party is not available. To address this problem, we propose a novel approach that allows each party to jointly conduct local atomicity checking with its direct partners. It is based on iterative forwarding and regression of compensability properties between each pair of direct partners. This approach is applied to a case study based on a real-life insurance process in the motor damage claims domain.

Categories and Subject Descriptors

H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces -*Theory and models, Web-based interaction*

General Terms

Algorithms, Design, Theory, Verification.

Keywords

Process collaboration, atomicity, process algebra, privacy.

1. INTRODUCTION

It is vital to respect the privacy of individual process collaborators in a Business-to-Business (B2B) environment, where collaborating business processes are typically distributed, autonomous, heterogeneous and enacted individually. To retain business interest or privacy, organizations only expose their restricted public views, hiding the internal business logistics (or internal processes) from others. The collaboration of these processes is through message exchanges between their public views [3][4][17]. To handle the inconsistency problem in process collaboration, weak consistency approaches, such as exception handling, are often adopted in such environments [10].

Exception handling approaches relax conventional database transaction requirements and impose *atomicity sphere* on business processes [10]. Satisfaction of atomicity sphere is a desirable approach to guarantee atomicity property in a business process. If a business process *satisfies* atomicity sphere, it should warrant the business process to terminate with the semantics of all or nothing. Such a business process can always terminate successfully by retrying the failed tasks, or rolling back to its original state without undesirable side effects by compensating all the executed tasks. In contrast, the abortion of a business process that does not exhibit atomicity sphere may suffer from side effects and result in value loss, e.g., payment cannot be refunded when a procurement transaction is aborted.

Even if atomicity sphere property is held in every collaborating process individually, this property may still not be respected in their collaboration. It is therefore desirable to check in advance to determine whether the atomicity sphere property may be violated in a potential B2B process collaboration. The global analysis approach, which checks atomicity sphere by composing all the collaborating business processes into a composite process, is not applicable in some scenarios when the business parties refuse to expose their internal business processes due to business or privacy reasons. Our earlier work [21] has solved this problem by proposing a novel process algebraic model to derive atomicity-equivalent public views from detailed business processes. By checking the atomicity sphere in the composition of these public views, the atomicity property of their collaboration can be verified. It exempts business parties from releasing the details of its business processes to its partners while checking the atomicity sphere of their collaboration. Thus, such a checking requires a trustworthy party who has complete knowledge of the public views that forms the composite process.

Many organizations may hesitate to take avoidable risks and conduct potentially problematic business transactions. Therefore, how to verify atomicity sphere of B2B collaborations in time when the trustworthy party is unavailable, even temporarily? Let us illustrate this problem using an example. To ease the

* This research is supported by a grant from the Research Grants Council of Hong Kong (Project No. HKUST6167/04E).

§ All correspondence should be addressed to Dr. S.C. Cheung at Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China. Tel: (+852) 2358-7016. Fax: (+852) 2358-1477. Email: scc@cse.ust.hk.

♣ Parts of the work were done while W.K. Chan was with the Hong Kong University of Science and Technology before August 2006.

discussion, we refer to a direct business collaborator of an organization as a *direct partner* of the organization. Consider the scenario involving a retailer, a supplier, a bank, a producer, and a shipper. Suppose that they collaboratively provide a supply chain application, and a trustworthy party does not exist in such scenario because each party wants to hide their collaboration information from others due to privacy or business reasons. For instance, from the supplier's perspective, its customer (i.e., the retailer) should better be blind to observe its vendors (i.e., the producer) and on how they collaborate. Similarly, from the perspective of the retailer, its suppliers should not know its interactions with its clearance bank. Hence, the global checking approach for atomicity sphere [21] is inapplicable in this scenario.

To solve the problem, we propose a local analysis approach to verify atomicity sphere of B2B process collaboration. Instead of composing all the public views and checking the atomicity sphere by a trustworthy party, each business party in this approach verifies locally whether the collaboration with its direct partners satisfies atomicity sphere. In this way, the mandate for a trustworthy party is not required. Our model assumes that each party may publish different public views for its different direct partners. We call them *local views*. Each local view of a business process is restricted to contain the essential information directly relevant to a particular direct partner of the business process. Thus, this approach avoids disclosing information amongst direct concerned partners.

The main challenge of the proposed approach lies in the problem of analyzing atomicity sphere accurately without global knowledge of all involved parties. Reasoning the atomicity sphere of a B2B collaboration requires some global knowledge of process interaction, including the partial ordering of all tasks, and their compensability and retriability properties. We will discuss them further in Section 2. These kinds of information, however, are unavailable or only partially available in each organization. Hence, it is difficult to conduct accurate analysis of atomicity sphere of their B2B collaboration locally by each involved business party. We will elaborate on this problem and examine various inaccuracies that may arise in Section 3. Currently, methods to uncover the global knowledge required for the checking of global atomicity in local analysis has not been addressed in existing related research work.

The main contributions of this paper are twofold: First, a novel analysis approach based on property propagation on local views is proposed to confine the potential violation of atomicity sphere within individual processes. By analyzing a local atomicity sphere criterion of every involved process, business parties as a whole could identify whether the atomicity property is achieved in their collaboration. Second, algorithms are presented to construct local views, propagate property amongst business parties, and check the local atomicity sphere criterion.

The rest of this paper is organized as follows: Section 2 reviews the background of a process algebraic model for business processes, and global checking of atomicity sphere. Section 3 analyzes the problems in local checking. Section 4 gives our solution and theoretical results. Section 5 illustrates this proposal by using an insurance case study based on motor damage claims. Section 6 presents the evaluation and discussions of this proposal. Section 7 summarizes the related work and draws a comparison. This is followed by our conclusions in Section 8.

2. BACKGROUND

2.1 Business process algebraic model

In this section, we will review the definition of business process and the atomicity-equivalent process algebra PA^a [21]. We denote the termination state by the special constant *0*.

Definition 1: A process p is defined by a quadruple (p, P, A, F), where P is a set of states, A is a set of actions, $F \subseteq (P \times A \times P)$ is a ternary transition relation. Note that $0 \in P$. An *action* in this paper refers to the commitment of an executing task. Actions occur instantaneously. For simplicity, we denote the initial state of a process p as p, and p^* as all the states reachable from p. We assume that a B2B process always terminates [14].

Definition 2: Several basic operators are defined for processes with the precedence "·", "+", "‖".

"·": a prefix operator on processes
"+": a choice composition operator on processes
"‖": a parallel composition operator on processes

In a process, a *port action* is designed to communicate with other processes. A process p with a set of port actions H is denoted as p_H. *Non-port actions* do not participate in communications with other processes. For simplicity, we assume that two port actions from different processes with the same name indicate that they communicate with each other synchronously. We also assume that non-port actions (except silent action, c.f., Section 4.2) in different processes do not share the same name. The following axioms describe the behavior of the three basic operators "·", "+" and "‖". We denote a process algebra system with the following axioms as *BPA* (short for basic process algebra).

Let P be a set of processes. $\forall x, y, z \in P, a, b \in A.$

$x + y = y + x$	*(A1)*	$(x + y) + z = x + (y + z)$	*(A2)*
$x + x = x$	*(A3)*	$(a + b) \cdot z = a \cdot z + b \cdot z$	*(A4)*
$x + 0 = x$	*(A5)*	$x \parallel y = y \parallel x$	*(M1)*
$(x+y) \parallel z = x \parallel z + y \parallel z$	*(M2)*	$0_{H1} \parallel 0_{H2} = 0_{H1 \cup H2}$	*(M3)*

$a \cdot x_{H1} \parallel 0_{H2} = a \cdot (x_{H1} \parallel 0_{H2})\ (a \notin H_1 \cap H_2)$ *(M4)*
$a \cdot x_{H1} \parallel 0_{H2} = 0_{H1 \cup H2}\ (a \in H_1 \cap H_2)$ *(M5)*
$a \cdot x_{H1} \parallel b \cdot y_{H2} = a \cdot (x_{H1} \parallel b \cdot y_{H2}) + b \cdot (y_{H2} \parallel a \cdot x_{H1})\ (a, b \notin H_1 \cap H_2)$ *(M6)*
$a \cdot x_{H1} \parallel b \cdot y_{H2} = a \cdot (x_{H1} \parallel b \cdot y_{H2})\ (a \notin H_1 \cap H_2, b \in H_1 \cap H_2)$ *(M7)*
$a \cdot x_{H1} \parallel b \cdot y_{H2} = a \cdot (x_{H1} \parallel y_{H2})\ (a, b \in H_1 \cap H_2, a = b)$ *(M8)*
$a \cdot x_{H1} \parallel b \cdot y_{H2} = 0_{H1 \cup H2}\ (a, b \in H_1 \cap H_2, a \neq b)$ *(M9)*

The atomicity-equivalent process algebra PA^a is an extension to the *BPA* with some extra simplification rules for silent actions and violation of atomicity sphere. Details could be found in [21], and this paper does not rely on them. In this algebra, we denote $PA^a \vdash p_1 \rightarrow p_2$ if p_1 can be reduced to p_2 using the axioms of PA^a.

Definition 3: A complete trace of a process is a sequence, possibly empty, of actions executed from its initial state to its termination state. For a process p, *trace*(p) denotes the set of all the complete traces of process p. For a trace t, i is an integer larger than 0, $t[i]$ denotes the action in the ith position of t. Let A_i be a set of actions, $t \backslash A_i$ represent the trace removing all the actions except those in A_i.

2.2 Global checking of atomicity sphere

In a business process, each task is assigned with two properties: *compensability* and *retriability* [10]. A compensable task can be undone one way or the other, whenever the process fails or is aborted. A task is *non-compensable*, referred to *as a pivot task*, if the overhead or cost of compensating the task is unacceptable [14].

A retriable task can, without cumulative effects, repeat itself internally if the latest internal trial fails, and finally succeed after a finite number of trials; otherwise, a task is non-retriable. In our model, every action $a \in A$ has two essential properties: *compensability* and *retriability*. Predicate C(a) is true if and only if the action a is compensable; predicate R(a) is true if and only if the action a is retriable. Note that the semantics of "atomicity" used in business processes is different from that used in database in the sense that "the structure [business process] guarantees a more general notion of atomicity than traditional transactions" [14]. Informally, a process in this model satisfies atomicity sphere if and only if (i) it does not contain any state that violates atomicity sphere (denoted as ϕ), and (ii) no non-retriable task is executed after a non-compensable task in any of its complete execution traces. This is formalized as follows:

Definition 4: φ is a predicate of a process, where $\varphi(p)$ is true if and only if the following conditions hold:

$\phi \notin p^*$

$\neg\exists t \in trace(p)\ [(a=t[i], b=t[j], j>i>0) \wedge (C(a)=false, R(b)=false)]$.

Otherwise, $\varphi(p)$ is false. For any processes p_1, p_2, if $\varphi(p_1) = \varphi(p_2)$, p_1 and p_2 are said to be *atomicity-equivalent*.

In the global checking of atomicity sphere, atomicity-equivalent public views are derived from collaborative processes, and then composed into a global process (we refer to the composition of all the collaborative processes or their atomicity-equivalent public views as a *global process,* and each collaborative process as a *private process*). Their collaboration satisfies atomicity sphere if and only if the global process satisfies atomicity sphere. Formal details can be found in [21]. In the subsequent discussion, we assume that each business party owns one private process. A party that owns multiple private processes can compose them into one using the process algebraic model presented previously.

3. PROBLEMS AND CHALLENGES

As mentioned in Section 1, the local checking approach has potential to alleviate the privacy problem. Still, it may suffer from inaccuracy due to the lacking in global (complete) knowledge about the process collaboration. In general, there are two kinds of inaccuracy that could be introduced by a local checking approach, namely, *under-estimation* and *over-estimation*.

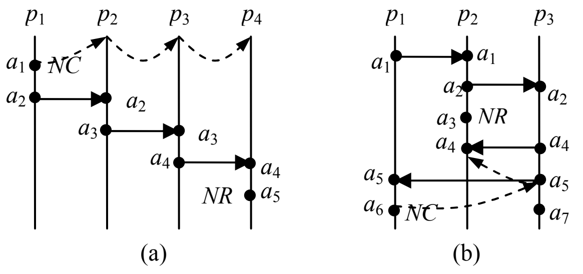

Figure 1: Inaccuracy in local checking.

Problem 1 (Under-estimation inaccuracy): This inaccuracy occurs when there is a scenario where the collaboration of business processes violates atomicity sphere, but the local checking shows that it satisfies. This kind of inaccuracy will introduce value loss for process collaboration because they cannot exclude the violation of atomicity sphere in advance. One of the reasons is due to the losing track of some pivot tasks of other processes in the local checking. For example, as depicted in Figure 1(a), processes p_1, p_2, p_3, and p_4 collaborate amongst themselves. Tasks are represented as black circles, and the solid arrow between two tasks in different processes represents their synchronous communication (the dashed arrow represents property propagation, c.f., Section 4.1). Task a_1 in process p_1 is non-compensable, task a_5 in p_4 is non-retriable, and the other tasks are compensable and retriable. One could observe that the collaboration of each business party with its direct partner(s) satisfies atomicity sphere. However, their overall collaboration violates atomicity sphere because non-retriable task a_5 is executed after non-compensable task a_1.

Problem 2 (Over-estimation inaccuracy): Over-estimation occurs when the collaboration of business processes satisfies atomicity sphere, but the local checking returns false report. Over-estimation inaccuracy will potentially exclude legitimate collaborative services and hence limit the choices for selection. A reason for the local checking to report a false alarm is that some temporal relationships amongst tasks in different processes may not be accessible during local checking using existing techniques. Consider the example in Figure 1(b). The collaboration of p_1, p_2, and p_3 satisfies atomicity sphere, in which task a_3 is compensable and non-retriable, a_6 is non-compensable and retriable, and the others are compensable and retriable. The local checking of p_2 reports a violation because the information of the temporal relationship between tasks a_3 and a_6 are not available for the checking because the collaboration of p_1 and p_3 (that is, a_5 in p_1 communicates with a_5 in p_3) is invisible to p_2 due to privacy. Hence, from the perspective of p_2, the non-compensable task a_6 could be executed before the non-retriable task a_3 and thus appears to violate atomicity sphere property.

Both kinds of inaccurate results are undesirable in B2B collaboration. However, these problems cannot be solved by existing approaches [10][21] The challenge lies in that the information sharing between business parties in these approaches will unavoidably expose their privacy, while the protection of privacy will compromise the global knowledge of process interaction, which is critical to the accuracy of the final analysis results. We have studied this problem closely and found that a global knowledge of process interaction is not needed for the reasoning of atomicity sphere if business parties are willing to let their direct partners know when their process becomes non-compensable. We will discuss this assumption later.

In the next section, we will propose an approach based on the propagation of non-compensable properties to solve these two inaccuracy problems.

4. LOCAL ANALYSIS

4.1 Overview

To tackle the two inaccuracy problems presented in Section 3, we propose a property propagation technique. We observe that if a process can transfer the relative positions of its pivot tasks to a destination process, the latter can judge whether its tasks may fail atomicity sphere. However, such propagation of pivot tasks from one process to another may expose the privacy. A solution is to propagate only the non-compensable property of pivot tasks instead of the tasks themselves. For example, as depicted in Figure 1(a), the non-compensable property of pivot task a_1 in process p_1 can be propagated to the start position of p_2 (a dashed arrow represents the property propagation direction). The intuition of such propagation is that a_1 can be executed before the first task of p_2 in their collaboration. Thus the propagation reveals to p_2 that some non-compensable task in p_1 could be executed before any task of p_2. This non-compensable property can then be propagated to p_3 and p_4 in similar ways. Thus, process p_4 knows that there is some non-compensable task in some other process which may be executed before its non-retriable task a_5. Hence, p_4 can determine locally that their collaboration violates atomicity sphere. Likewise, in Figure 1(b), the non-compensable property of pivot task a_6 can be propagated to p_3 at the point right after a_5. Its intuitive meaning is that task a_6 cannot be executed before a_5, but can be executed before any of a_5's subsequent tasks in p_3. The subsequent propagation of this property from p_3 to p_2 at the point after a_4 explores to p_2 that the pivot task a_6 is always executed after a_4. Therefore, p_2 knows that a_6 is always executed after a_3 so that the local checking of p_2 does not give a false report.

The above observation has inspired us to propose an approach based on property propagation of pivot tasks to analyze atomicity sphere locally. The basic idea is as follows: First, it confines the potential violation of atomicity sphere of the global process to individual private processes by propagating the non-compensable property of some pivot tasks to all the partners. Next, by checking a local atomicity sphere criterion of each individual processes, business parties can identify whether their collaboration satisfies atomicity sphere.

In the next section, we will formalize the property propagation approach and prove that it is sound for checking the satisfaction of atomicity sphere locally.

4.2 Property propagation

Before introducing the property propagation approach, we will first introduce some essential definitions.

Definition 5: A *primary pivot task* of a process *p* is the first non-compensable task in a complete trace of *p*. Let *ppt*(*a*) be a predicate describing whether *a* is a primary pivot task. That is, $ppt(a)=true$ *iff* $\exists t\in trace(p)$, $a=t[i]$, $C(a)=false$, $\wedge$ $(\forall 0<j<i$, $C(t[j])=true)$.

For example, as depicted in Figure 1(b), task a_6 in p_1 is a primary pivot task because it is the first non-compensable pivot task in the complete trace of p_1. Similarly, task a_1 in Figure 1(a) is also a primary pivot task.

We observe that a process violates atomicity sphere if and only if some non-retriable task is executed after a non-compensable task in one of its complete traces [21]. We therefore consider only propagating the non-compensable property of primary pivot tasks in our approach, because these kinds of information, as we will prove later, is enough to locate the violation of atomicity sphere of the global process by individual private processes. To ease discussions, in the remaining of this paper, we refer to *property propagation* as the propagation of non-compensable property of primary pivot tasks from one process to the others.

To avoid revealing pivot tasks, an operator Υ is introduced to project the private process onto its local views, where non-port pivot tasks are abstracted as silent actions, denoted as τ. Business parties can then use these local views to exchange the points where non-compensable properties hold. Since τ is used to abstract pivot points where non-compensable properties hold, it is considered to be non-compensable in local views. Meanwhile we need to propagate only the non-compensability properties as shown by a proof later, τ is thus assigned to be retriable.

The operator Υ performs two operations: (i) renames each primary pivot task to τ if the primary pivot task is not a port action in the local view, and (ii) discards all the other tasks in the process except the port actions required in the local view. Each process may have its own primary pivot tasks. To differentiate the properties of different primary pivot tasks amongst different processes, we assign each primary pivot task α an identity (ID) set, represented by $\alpha.idset$, which contains a unique identification, denoted by $\alpha.id$, that is $\alpha.idset=\{\alpha.id\}$. Similarly, each silent action τ owns an ID set, which contains the IDs of the primary pivot tasks that have been renamed to this silent action. Note that the ID set of one silent action can differ from that of another silent action. To ease the presentation, we use predicate *silent*(*a*) = *true* to represent that *a* is a silent action τ.

***Definition 6* [Local View]**: $\Upsilon(p, H_i)$ is a projection operator that projects a process *p* onto a local view with the set of port actions H_i. It is defined inductively as follows:

1) $\Upsilon(0, H_i) = 0$

2) $\Upsilon(a\cdot x, H_i) = a \cdot \Upsilon(x, H_i)$, *if* $a\in H_i$ *or* $silent(a)=true$; *otherwise* $\tau \cdot \Upsilon(x, H_i)$, *where* $\tau.idset= \{a.id\}$, *if* $ppt(a) = true$, $\Upsilon(x, H_i)$, *if* $ppt(a) = false$.

3) $\Upsilon(x+y, H_i) = \Upsilon(x, H_i) + \Upsilon(y, H_i)$

Each private process may have multiple distinct local views, each of which is constructed for the interaction with one of its direct partners. The local view contains only the required information for such interaction and the associated property propagation. For example, the process p_1 in Figure 1(b) can be represented as $a_1\cdot a_5\cdot a_6\cdot 0$, where a_6 is a primary pivot task. Process p_1 collaborates with p_2 via the set of port actions $H=\{a_1\}$. Using the rules in Definition 6, one can derive that p_1's local view for p_2 is $a_1\cdot \tau \cdot 0$. Similarly, its local view for p_3 is $a_5\cdot \tau \cdot 0$.

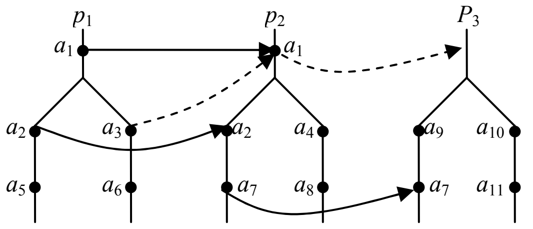

Figure 2. Silent action with excluded set.

We also associate every silent action with a powerset of port actions, which could not appear in the same trace with the silent action. We call it an *excluded set* of the silent action. Let us consider the illustration depicted in Figure 2. Suppose that the task a_3 is a primary pivot task of the process p_1, and it is propagated to the process p_2 at the point after a_1. Apparently, from the perspective of p_2, the propagated silent action of a_3 will execute before the port action a_2. However, from the process behavior of p_1, a_2 and a_3 cannot both appear in the same trace. Hence, to reflect such information in the process p_1, the corresponding silent action of a_3 is assigned an excluded set with its ID in the local view (that is, $(a_3.id, \{a_2\})$). When the silent action is propagated to p_2, the excluded set could be used to judge whether a port action should occur in the same trace with this silent action. To further propagate this silent action to the process p_3, our proposal is to re-calculate the excluded set by considering the local view for p_3, in the same manner as we propagate the silent action from p_1 to p_2. For example, since a_2 is in the excluded set of this silent action, a_7 is thus in its excluded set in p_2's local view for p_3.

A local view may contain multiple silent actions τ. To facilitate property propagation, a simplification rule is provided to merge two consecutive silent actions τ into one. We refer a local view that cannot be applied the simplification rule any more as *a canonical local view*. Note that the excluded set for each ID in the two silent actions is also merged after simplification.

$$\text{If } silent(a)=true,\ silent(b)=true, \text{ then } a \cdot b \cdot x = c \cdot x,$$
$$\text{where } silent(c) \wedge c.idset = a.idset \cup b.idset \qquad (S1)$$

We refer to the process algebraic model extending PA^a with τ and this simplification rule as PA^p. Under this extended framework, we introduce an operator to propagate properties of primary pivot tasks. Such propagation inserts the silent action τ (that is, the non-compensable property of primary pivot tasks) from a local view into a private process at its earliest position where those corresponding renamed primary pivot tasks of this silent action τ may occur. Its formal definition is given below.

***Definition** 7* **[Property propagation]**: Let p be a process that collaborates with another process q using the set of port actions H, and that qv is the local view projected from q for the collaboration with p, that is, $PA^p \vdash \Upsilon(q, H) \rightarrow qv$. The property propagation from qv to p generates a new process p', denoted as $p'=\lambda_H(p, qv)$, which is defined inductively by the following rules:

1) $\lambda_H(x, 0) = 0$ 2) $\lambda_H(0, b \cdot x) = 0$ *if* $b \in H$
3) $\lambda_H(0, \tau \cdot x) = \lambda_H(\tau \cdot 0, x)$ 4) $\lambda_H(x+y, z) = \lambda_H(x, z) + \lambda_H(y, z)$
5) $\lambda_H(a \cdot x, b \cdot y) = a \cdot \lambda_H(x, y)$ *if* $a, b \in H \wedge a=b$
6) $\lambda_H(a \cdot x, b \cdot y) = a \cdot \lambda_H(x, b \cdot y)$ *if* $a \notin H \wedge b \in H$
7) $\lambda_H(a \cdot x, b \cdot y) = a \cdot x$ *if* $a, b \in H \wedge a \neq b$
8) $\lambda_H(a \cdot x, \tau \cdot y) = \lambda_H(\tau \cdot a \cdot x, y)$ *if* $silent(a)=false$
9) $\lambda_H(a \cdot x, b \cdot y) = \lambda_H(c \cdot x, y)$ *where* $c.idset = a.idset \cup b.idset, \wedge$ $silent(a)=silent(b)=silent(c)=true$
10) $\lambda_H(x, y+z) = \lambda_H(\lambda_H(x, y), z)$

The intuition of these rules is to insert silent action τ from its direct partner's local view qv into the process p at the earliest point it can occur. In particular, rules 3 and 8 mean that the silent action is inserted into the private process at its earliest point. Rule 9 represents that propagations of different non-compensable properties to the same position of a private process are converged. Rule 10 represents that the propagation of two alternative views could be propagated one by one. For example, as depicted in Figure 1(b), a_6 is a primary pivot task of p_1, and p_1 is a direct partner of p_3. The local view of p_1 for its direct partner p_3 is $a_5 \cdot \tau \cdot 0$. Meanwhile, p_3 can be represented as $a_2 \cdot a_4 \cdot a_5 \cdot a_7 \cdot 0$. Applying the rules of Definition 7, the propagation of property from a_6 to p_3 returns the result $a_2 \cdot a_4 \cdot a_5 \cdot \tau \cdot a_7 \cdot 0$, which inserts the silent action τ into the point after a_5 in p_3.

To decentralize the global checking of atomicity sphere satisfaction to individual parties, we need to recover the partial ordering between primary pivot tasks and non-retriable tasks across all participating processes when they interact. Such partial ordering can be derived by applying iteratively the property propagation between all collaborating private process pairs. We provide an algorithm to propagate non-compensable properties of primary pivot tasks as follows, which can guarantee that the non-compensable property of each primary pivot task could be propagated to all of its direct and indirect partners.

Algorithm 1 (Propagation of non-compensable property)

```
1. Mark primary pivot tasks in each private process p.
2. For p's each direct partner $q_i$, which collaborates with p with the set of port action $H_i$, project p onto a corresponding local view $pv_i$ (with respect to $H_i$), that is, $PA^p \vdash \Upsilon(p, H_i) \rightarrow pv_i$.
3. For p's each direct partner $q_i$, which collaborates with p with the set of port actions $H_i$, let $qv_i$ be its latest updated and canonical local view, that is, $PA^p \vdash \Upsilon(q_i, H_i) \rightarrow qv_i$, update the process p by $p=\lambda_{H_i}(p, qv_i)$.
4. Repeat steps 2 and 3 until all the private processes do not change any more.
```

This algorithm exchanges non-compensable properties of primary pivot tasks between direct business partners via their local views. In step 3, for the purpose of atomicity sphere checking, the algorithm updates a private process using the local views of its direct partners, and then updates its local views in step 2 accordingly. Therefore, the non-compensable property of primary pivot tasks could be exchanged between its partners. For example, as depicted in Figure 1(a), the non-compensable property of the primary pivot task a_1 is first propagated to p_2 from p_1, and then it is further propagated to p_3, p_4 via p_2. If there are n processes in a B2B collaboration, after a finite number (at most n) of steps, the non-compensable property of each primary pivot task is propagated to all the involved parties via all possible propagation paths. This is formalized in the following theorem.

Theorem 1:
1) Algorithm 1 always terminates.
2) When Algorithm 1 terminates, the non-compensable property of each primary pivot task is propagated to all other private processes via all possible propagation paths.

It is easy to observe that the number of primary pivot tasks is finite, and the number of transitions in each private process during property propagation is finite. Meanwhile, the non-compensable property of a primary pivot task (that is, τ) can only be inserted between two non-silent transitions of a private process. Thus, after inserting a finite number of non-compensable properties in each private process, this insertion step will reach a fixed point and terminate. Suppose that Algorithm 1 has terminated and there is still a non-compensable property of a primary pivot task that has not been propagated to some private process via some path. The propagation related to this private process would not reach any fixed point because this process could still be changed by inserting a new silent action τ. This contradicts the fact that Algorithm 1 has terminated. The formal proof of this theorem can be found in [22].

Algorithm 1 is to be executed concurrently by individual business parties. We need a mechanism to detect the fixed point state where further execution of the algorithm by any parties does not derive new silent actions in local views. A solution is to deploy a distributed consensus protocol such as the one proposed in [6] so that each party can confirm with its direct partners if they can no longer find new silent actions. Owing to space limitation, we will not cover the protocol here.

In the next section, we will introduce a local criterion to analyze the atomicity sphere locally. By checking the local atomicity sphere criterion of each private process, we can identify whether their collaboration satisfies the overall atomicity sphere or not.

4.3 Local analysis of atomicity sphere

One problem in local analysis is that the non-compensable property of a primary pivot task may be propagated to a point in the private process that is not reachable in the global process, or the primary pivot task itself is not reachable in the global process due to deadlock or other reasons. For example, as depicted in Figure 3, Task a_5 in p_1 is a primary pivot task, and its property is propagated to p_2 at the point after task a_2. After the execution of task a_1, p_2 may choose to execute either task a_2 or a_3. Task a_6 in p_2 is a non-retriable task, and all other tasks are compensable and retriable. It is easy to observe that a_5 is not reachable in the global process due to communication deadlock. However, its property is propagated to p_2, which has no knowledge that a_5 is not reachable in the global process.

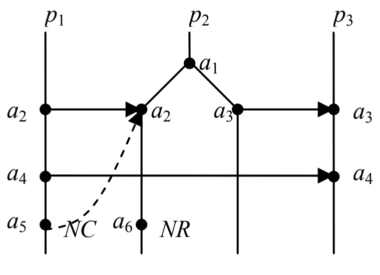

Figure 3: Reachability of primary pivot task.

From the perspective of business collaboration, deadlock is not desirable because it may lead to abnormal termination of business process execution in some scenarios. Efficient techniques of deadlock detection and reachability analysis for distributed communicating processes are available [1]. Hence, before applying our property propagation approach, business parties could apply these techniques [1] to check the existence of communication deadlocks and to remove non-executable transitions in their collaboration. Hence, we restrict ourselves to consider a restricted subclass of B2B processes which are *well-formed*, that is, deadlock-free, and all transitions are reachable in the global process. We apply our approach to the well-formed processes, whose behaviors can be expressed in tree-like structures, where transitions are uniquely labeled. We will discuss the scalability issue in Section 6. The following definition describes this subclass of business processes.

Definition 8 **[well-formedness]**: Suppose business processes p_i= (s_{i0}, S_i, A_i, F_i), i=1..n, collaborate with one another. They are *well-formed* if and only if the following conditions are satisfied:

(1) $\forall(s_{ij}, a, s_{ik}), (s_{il}, b, s_{im}) \in F_i$: *if* $s_{ik} = s_{im}$ *then* $s_{ij} = s_{il} \wedge a=b$, *otherwise*, $a \neq b$.

(2) $\forall t \in trace(p_1 \| p_2 \| \ldots \| p_n)$, $t \backslash A_i$ *is empty or a complete trace of* p_i.

(3) $\forall t \in trace(p_i)$, $a=t[l]$, $b=t[k]$, $k \geq l > 0$: $\exists t' \in trace(p_1 \| p_2 \| \ldots \| p_n) \wedge k' \geq l' > 0$, $t'[l']=a$, $t'[k']=b$.

Since the non-compensable property of a primary pivot task may be propagated to the same private process via different paths, they may not propagate to the same position. This results in inaccuracy, because during the propagation, some partial order relationships between actions in different processes could be lost if they are not well-controlled. For example, as depicted in Figure 4, a_6 is a primary pivot task, and its non-compensable property is propagated to p_4 via different paths: One silent action is inserted after a_1, and another is inserted after a_3. According to the checking criterion for atomicity sphere [21], the private process p_4 will violate atomicity sphere after the property propagation of primary pivot task a_6. However, from the perspective of the global process, a_6 could not appear at the point in p_4 after a_1 in the global process.

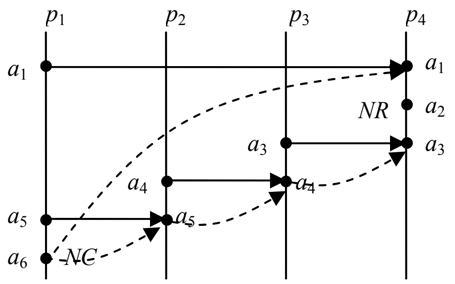

Figure 4: Duplicate non-compensable property via different propagation paths.

To solve this problem, we propose a novel criterion for checking atomicity sphere of individual private processes after the property propagation. Recall that each transition has a unique action, and, to ease the discussion, we use the symbol "$a \Rightarrow b$" to represent that action a is executed before b in some trace of a private process. If a is unable to execute before b in any trace of a private process, we denote it as "$\neg(a \Rightarrow b)$". We define that the predicate *excluded*(a, b) is *true* if and only if *silent*(a)=*true* and there exists a port action c in the excluded set of a having c=b or c$\Rightarrow$ b.

Definition 9 **[Local atomicity sphere criterion]**: η is a local property of process p = (p, S, A, F), $\eta(p)$ = *true* if and only if $\forall a, b \in A$: if $a \Rightarrow b \wedge R(b)$=*false*, then at least one of the following conditions is satisfied:

(*i*) $C(a)$=*true*,

(*ii*) *silent*(a)=*true* $\wedge\ \exists c \in A$: $a \Rightarrow c \wedge \neg(b \Rightarrow c) \wedge a.idset \subseteq c.idset$, *or*

(*iii*) *silent*(a)=*true* $\wedge$ *excluded*(a, b)=*true*

The intuitive meaning of condition (*i*) is that there is no non-retriable task executed after some non-compensable task in some trace of this process. Condition (*ii*) is to remove the duplicate silent actions propagated from the same primary pivot task. If one silent action is executed before a non-retriable task in some trace, but there exists a duplicate silent action from the same primary pivot task, which is executed after the first silent action in some trace but not before this non-retriable task in some trace, then the effects of the first silent action is removed. For example, as depicted in Figure 4, the silent action inserted after a_1 and the one inserted after a_3 in p_4 are both from a_6. The first silent action is executed before the non-retriable task a_2, but the effect of this silent action is removed because the second silent action is executed after the first one but not before a_2 in some trace. Condition (*iii*) says that b is not in the same trace with the primary pivot task. According to this criterion, business parties could check whether their collaboration satisfies atomicity sphere locally. This can be formalized by the following theorem:

Theorem 2: Suppose well-formed processes p_1, p_2, …, and p_n collaborate with one another. Let p_1', p_2',…, and p_n' be the corresponding processes after Algorithm1 terminates, then we have $\eta(p_1') \wedge \eta(p_2') \wedge \ldots \wedge \eta(p_n') = \varphi(p_1 \| p_2 \| \ldots \| p_n)$.

Proof: If $\eta(p_1') \wedge \eta(p_2') \wedge \ldots \wedge \eta(p_n')$ = *true*, but $\varphi(p_1 \| p_2 \| \ldots \| p_n)$ = *false*, then there exist at least two tasks a and b in the process $p_1 \| p_2 \| \ldots \| p_n$, where $ppt(a)$=*true*, $R(b)$=*false*, and a is executed before b in some complete trace of $p_1 \| p_2 \| \ldots \| p_n$. Suppose a and b belong to the same process p_i, then we have $\eta(p_i')$ = *false*. On the other hand, suppose a belongs to p_i, and b belongs to p_j ($i \neq j$), and the non-compensable property of a is propagated to p_j'. Since $\eta(p_j')$ = *true*, by exhaustion, there are only three cases for the corresponding silent action (represented by c_1) of primary pivot task a and b in any complete trace of p_j': Case(*i*), $c_1 \Rightarrow b$, but there exists another corresponding silent action (represented by c_2) satisfying $c_1 \Rightarrow c_2 \wedge \neg(c_2 \Rightarrow b) \wedge c_1.idset \subseteq c_2.idset$. Suppose c_2 is inserted after the port action d (or as the initial action), then we also have $\neg(d \Rightarrow b)$. Since the processes are well-formed, so a can be executed in the global process only after d is executed. Hence, a could not be executed before b in any complete trace of $p_1 \| p_2 \| \ldots \| p_n$; Case(*ii*), $c_1 \Rightarrow b \wedge excluded(c_1, b)$= *true*, or Case(*iii*), $\neg(c_1 \Rightarrow b)$. These two cases also contradict the assumption that a is executed before b in some complete trace of $p_1 \| p_2 \| \ldots \| p_n$. Hence, If $\eta(p_1') \wedge \eta(p_2') \wedge \ldots \wedge \eta(p_n')$ = *true*, then $\varphi(p_1 \| p_2 \| \ldots \| p_n)$ = *true*.

On the other hand, suppose $\varphi(p_1 \| p_2 \| \ldots \| p_n)$ = *true*, but $\eta(p_1') \wedge \eta(p_2') \wedge \ldots \wedge \eta(p_n')$ = *false*, then there are two cases: Case(*i*), $\varphi(p_j)$=*false*, then we have $\varphi(p_1 \| p_2 \| \ldots \| p_n)$ = *false* since the processes are well-formed; Case (*ii*), $\eta(p_j')$=*false* is caused by the propagation of the non-compensable property of a primary pivot task a from process p_i to p_j, such that the corresponding silent action (represented by c_1) is executed before a non-retriable task b. Since $\varphi(p_1 \| p_2 \| \ldots \| p_n)$=*true*, and the processes are well-formed, in every execution path of process $p_1 \| p_2 \| \ldots \| p_n$, b is executed before a; otherwise a and b should not be on the same trace. If b is executed before a in all the traces of the process $p_1 \| p_2 \| \ldots \| p_n$ that contains both a and b, then there exists a partial execution order between task b in p_j and task a in p_i, so that another corresponding silent action (represented by c_2) of the primary pivot task a could be propagated to p_j, and $b \Rightarrow c_2$, which contradicts that $\eta(p_j')$=*false*; If a, b are not included in the same execution trace of p_j', then either $excluded(c_1, b)$ = *true*, or there exists another silent action c_2 satisfying $c_1 \Rightarrow c_2 \wedge \neg(c_2 \Rightarrow b) \wedge c_1.idset \subseteq c_2.idset$, because the processes are well-formed. Obviously, these two cases also contradict the assumption that $\eta(p_j')$ = *false*. Hence, If $\varphi(p_1 \| p_2 \| \ldots \| p_n)$ = *true*, then $\eta(p_1') \wedge \eta(p_2') \wedge \ldots \wedge \eta(p_n')$ = *true*. □

This theorem shows that if all the private processes satisfy the local atomicity sphere criterion after the above property propagation, then their collaboration satisfies atomicity sphere; otherwise, if at least one local criterion η is violated after the non-compensable property propagation of primary pivot tasks, the collaboration of the well-formed collaborative processes will violate atomicity sphere. Based on Theorem 2, an organization can conclude locally whether its B2B collaboration with other business partners satisfy atomicity sphere.

4.4 Algorithms

In this section, we will refine the algorithm of propagating the non-compensable property from a local view to a private process and introduce an algorithm for checking local criterion. The realizations of Definitions 5 and 6 for the identification of primary pivot tasks and the projection of a private process onto a local view are relatively mechanical. We will not include the associated algorithms here due to page limitation. Details of these algorithms can be found in [22].

Let process $p=(s_{10}, P_1, A_1, F_1)$ collaborate with a process q with the set of port actions H, and $qv=(s_{20}, P_2, A_2, F_2)$ be q's local view for its partner p (that is, $PA^p \vdash \Upsilon(q, H) \rightarrow qv$). The following algorithm constructs $p'=(s, P, A, F)$ (that is, $p'=\lambda_H(p, qv)$). It propagates the non-compensable properties of primary pivot tasks of q to p via the local view qv. Note that the structure $(s_{1t}, s_{2k}, s_{1i}, b)$ is used in the algorithm, where s_{1t}, s_{2k} represent the current state of p and qv, respectively, and (s_{1i}, b, s_{1t}) is the last visited income transition of state s_{1t} (the initial state in line 1 has no visited income transition, represented as "_"). The Boolean variable *changed* is used to indicate whether the process p has been changed after the current property propagation. Lines 7-29 implement the rules of Definition 7.

Algorithm 2: (Property propagation)

```
1. $s \leftarrow \{(s_{10}, s_{20}, s_{10}, \_)\}$, stack.push(s), changed=false
2. for every $s_1 \in P_1$, $s_2 \in P_2$, visited($s_1$, $s_2$) ← false, endfor
3. while(not empty(stack))
4.    cs=($s_{1i}$, $s_{2j}$, ps, pa) ← stack.pop(), visited(($s_{1i}$, $s_{2j}$)) ←true
5.    for every ($s_{2j}$, a, $s_{2k}$)$\in F_2$
6.       for every ($s_{1i}$, b, $s_{1t}$)$\in F_1$
7.          if a=$\tau$,
8.             if b=$\tau$,
9.                if not(a.idset$\sqsubseteq$b.idset), changed←true, endif
10.                b.idset ← b.idset + a.idset,
11.               ns←($s_{1t}$, $s_{2k}$, $s_{1i}$, b)
12.               if not visited(($s_{1t}$, $s_{2k}$)), stack.push(ns), endif
13.            else
14.               ts←new state(), $F_1 \leftarrow F_1$ + {(ts, a, $s_{1i}$)}
15.               if ps = $s_{10}$, swap(ts, $s_{10}$), else
16.               $F_1 \leftarrow F_1$ + {(ps, pa, ts)}-{(ps, pa, $s_{1i}$)}endif
17.               changed←true, ns←($s_{1i}$, $s_{2k}$, ts, a)
18.               if not visited(($s_{1i}$, $s_{2k}$)),stack.push(ns), endif
19.            endif
20.         endif
21.         if a$\in$H
22.            if a=b
23.               ns←($s_{1t}$, $s_{2k}$, $s_{1i}$, b)
24.               if not visited(($s_{1t}$, $s_{2k}$)), stack.push(ns), endif
25.            endif
26.            if b$\notin$H
27.               ns←($s_{1t}$, $s_{2j}$, $s_{1i}$, b)
28.               if not visited(($s_{1t}$, $s_{2j}$)), stack.push(ns), endif
29.            endif
30.      endfor
31.   endfor
32. endwhile
```

Let p= (s, P, A, F) be a private process after applying the Algorithm 1. The algorithm for checking whether p holds the local criterion η is given as follows: The basic idea is to traverse the process from the termination state to the initial state to remove all the duplicate silent actions (Lines 1, 3-21). Then, the algorithm traverses the process in the reverse direction to check the

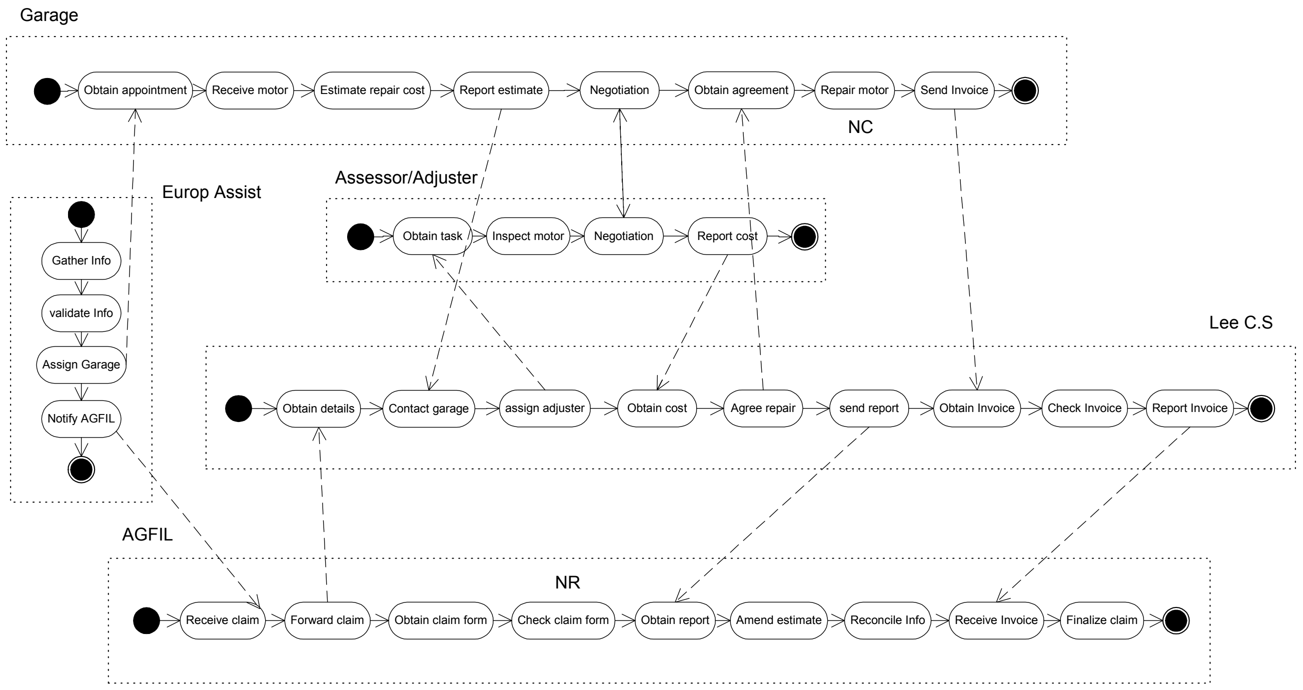

Figure 5: Process collaboration amongst multi-organizations.

atomicity sphere (Lines 2, 22-29). In Line 23, the predicate "*excluded*(*idset*, *a*)" returns true if and only there exists at least one ID in the *idset*, such that *a* is not in the excluded set of this ID. Due to space limitation, the calculation of the excluded set for an ID is not included in this paper, Details could be found in [22]. In Line 26, *NC* represents the set of non-compensable tasks of *p*.

Algorithm 3: (Check local criterion for atomicity sphere)

```
    1. removeduplicate(),
    2. check(s, {}).
removeduplicate():
    3. for every ts=0∈P,
    4.     idset ←{},
    5.     removereachable(ts, idset),
    6. endfor
removereachable(cs, set):
    7. if ∃(ps, a, cs)∈F,
    8.     if a=τ,
    9.         if a.idset ⊆ set,
    10.            F ← F - (ps, a, cs),
    11.            for every (cs, b, ns)∈F,
    12.                F ← F - (cs, b, ns) + (ps, b, ns),
    13.            endfor
    14.        else
    15.            a.idset← a.idset – set, set←set+ a.idset,
    16.        endif
    17.    else
    18.        if ppt(a)=true, set←set+ a.idset, endif
    19.    endif
    20.    removereachable(ps, set),
    21. endif
check(cs, idset):
    22. for every (cs, a, ns)∈F,
    23.     if R(a)=false ∧ idset ≠{}∧ ¬excluded (idset, a),
    24.         report violation, quit,
    25.     else
    26.         if a=τ ∨ a∈NC, idset←idset + a.idset, endif
    27.         check(ns, idset),
    28.     endif
    29. endfor
```

5. CASE STUDY

In this section, we illustrate our approach by using a real life business scenario depicted in Figure 5, in which "Garage", "Europ Assist", "Assessor/Adjuster", "Lee C.S", and "AGFIL" are all web services providers.

This case study scenario is derived from a real-life insurance example (motor damage claims) [5], in which AGF Irish Life Holding Plc. (AGFIL for short) is an insurance company operating in the Irish market providing domestic and commercial insurance. AGFIL collaborates with the other partners based on the operation of a scheme known as the Emergency Service, under which individual motor policyholders could claim back the repairing charges of their insurance-covered motor through the partners of AGFIL. The objective of this Emergency Service is to conduct damage claims in an efficient and cost effective manner.

In this scenario, Europ Assist provides a 24-hour emergency service hotline to individual motor policyholders. Lee C.S coordinates and manages the actual operation of the emergency service on behalf of AGFIL. It assigns Assessors/Adjusters to inspect the damage of a claimed motor and agree on the dollar amount for repairing if the prices do not deviate from the industrial norms. Garages provide repairing services to policyholders who do not need to pay the charges directly. Instead, the Garages forward the invoices to Lee C.S and receive the payment from AGFIL. AGFIL is responsible for underwriting the policies and covering the claimed losses. It does not take part in dollar amount negotiations, but it has the ultimate rights to decide whether a claim is faithful and whether any payment should be released to a claim. In case of an invalid claim (e.g., the loss is not covered in the policy terms), AGFIL will stop the claim procedure immediately.

Let us study their collaboration scenario, which is depicted in Figure 5. In the beginning, an individual motor policyholder makes a phone call to Europ Assist to initiate the damage claim procedure. Europ Assist then submits the claim form to AGFIL, who also forwards a copy to Lee C.S. On receiving the claim form, Lee C.S appoints some Loss Assessor/Adjuster to carry out the inspection of the claimed vehicle and negotiate the repairing price with Garage. At the same time, the AGFIL claim handler registers the claim and checks it manually. If Lee C.S agrees with the repairing price with a chosen garage, the garage will then commence motor repairing. Lee C.S will also issue AGFIL with an adjuster/assessors report. AGFIL will read the report and amend the estimated figures in the claims system. Finally, payments from AGFIL are made to the involved approved garages and assessors/adjusters by a third party payment system.

The collaboration of these processes provides an efficient service for motor policyholders. However, these organizations will suffer value losses in some scenarios. For example, if the damage claim is invalid (e.g., the loss is not covered in the policy terms), AGFIL will notify Lee C.S and stop the service for the motor policyholder concerned. However, the garage may have already repaired the motor. Hence, AGFIL or Lee C.S may also have to pay the garage the repairing charges, and thus suffer value losses.

One way to avoid this value loss scenario is to check their atomicity sphere before establishing their collaboration. This can be done by the global checking approach [21], in which an atomicity-equivalent public view is published by each involved process, and these public views are composed together into a global process. By checking atomicity sphere of this global process, these business parties could identify whether there exists some possible value loss scenarios in their collaboration.

However, this approach still exposes some privacy information of involved partners, because the atomicity-equivalent public view contains all the information about the collaboration of this process with all its direct partners. For example, from the perspective of AGFIL, it may not wish Garages and Europ Assist to know how it collaborates with Lee C.S due to its business privacy concerns. However, the atomicity-equivalent public view of AGFIL (c.f., Figure 6(a), where the "non-retriable silent action" is a kind of silent action defined in PA^a [21]) will expose such information. Furthermore, the composition of all the involved atomicity-equivalent public views will also reveal to Europ Assist that Lee C.S is a partner of AGFIL. The details of its collaboration with one of its partner are therefore known to its other partners.

The local analysis approach proposed in this paper can solve the above problems. Firstly, each private process publishes different local views for different partners. For example, as depicted in Figure 6(b), AGFIL publishes two local views, one for Europ Assist and another for Lee C.S. Each local view contains only essential information for process interaction and property propagation. Secondly, business parties exchange information only with its direct partners in the local analysis, and therefore do not disclose the partner information to unrelated business partners.

Before applying the local analysis approach, we apply the approach of [1] and find that they are well-formed. Then, based on Algorithm 1, the non-compensable property of the primary pivot task "repair motor" (since repairing motor takes time and other resources, it is non-compensable) in the garage process could be propagated to the AGFIL process at the point after task "Forward claim", as depicted in Figure 6(c). Meanwhile, the task "Check claim form" in the AGFIL process is non-retriable, because this task may raise an exception if the claim handler finds the claim is invalid, and thus stops the process. Hence, according to Definition 9, the AGFIL process violates the local criterion η after property propagation. According to Theorem 2, the collaboration of these processes violates atomicity sphere.

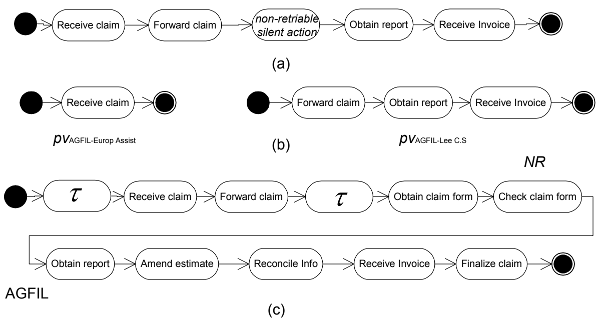

Figure 6: (a) Public view of AGFIL, (b) local views of AGFIL, and (c) AGFIL process after property propagation.

Based on the analysis results, business parties could observe the potential value loss scenarios in advance before their actual collaboration, and take measures to handle these scenarios (e.g., change the process, or choose other services and so on). This case study shows that the local analysis approach can also prevent value loss for B2B collaboration. Compared to the global checking approach, the local analysis approach does not disclose the collaboration information of one partner to the others, and is thus more applicable to the situations where a trustworthy party is not available.

6. EVALUATION

In this section, we will analyze the time complexity of the algorithms introduced in Section 4 to evaluate the feasibility of our proposal. This is followed by discussions regarding the limitations and some possible extension of our work.

6.1 Complexity analysis

6.1.1 Time cost for property propagation

Given process $p=(s_{10}, P_1, A_1, F_1)$, and q's local view $qv=(s_{20}, P_2, A_2, F_2)$ (i.e., $PA^p \vdash \Upsilon(q, H) \rightarrow qv$). Let $|F_1|=k_1$, $|F_2|=k_2$. For each transition of qv, the algorithm accesses at most k_1 transitions (Lines 5-31), so the total time cost is $O(k_1 \times k_2)$.

6.1.2 Time cost for checking local criterion

Given process $p=(s, P, A, F)$, let $|P| = m$, $|F|=k$. This algorithm accesses each state and transition at most twice, that is, traverse the process from the terminate state to initial state to remove duplicate silent actions (Lines 3-21), and traverse the process in the reverse direction to check atomicity sphere (Lines 22-29). The total time cost for this algorithm is thus $O(2m+2k)$. Since in a process, $m \leq k+1$, so the time cost for this algorithm is $O(k)$.

6.1.3 Time cost for Algorithm 1

Suppose there are n collaborative processes. For each business party, Algorithm1 will terminate after repeating at most n rounds.

Since the time cost for marking primary pivot task and constructing a local view is O(k) and O(k^2) [22], the time cost in each round is O($k^2 + m \times k \times k_{\max}$), where k, m, and k_{max} are the number of transitions of this process, the number of its direct partners ($m \leq n$), and the maximum number of transitions in its m partners' local views, respectively, so the total time cost is simplified as O($n \times k^2 + n^2 \times k \times k_{max}$). According to our investigation, the number of transitions in most existing business processes [13] ranges from 20 to 50. Hence, the approach proposed in this paper is applicable in practice.

6.2 Discussions

Our work assumes the availability of the non-compensable information of each business party to its direct partners. This is a win-win situation because the exchange of information about their tasks of no return could make business parties stay away from falling into an avoidable trap.

One challenge to deploy this approach in practice is that all involved parties are required to be trustworthy. This may not be true since they are autonomous and loosely coupled. During the property propagation, each party should keep the IDs of the silent action τ. If one party changes the IDs of a silent action or does not propagate it to others, then the final checking result might be inaccurate. We are investigating whether it is feasible to record the property propagation in the form of contracts. If their collaboration satisfies atomicity sphere in the checking but indeed incurs a value loss scenario at runtime, then business parties could trace the responsibility of the partner who does not propagate the silent action τ or change their IDs during the property propagation. We will investigate along this direction.

Another limitation is that our work requires the processes to be well-formed. However, existing works, such as [1], have provided approaches to detect deadlock and non-reachable transitions locally without compromising privacy. Examples on applying the approach [1] to detect deadlock and prune the processes to be well-formed could be found in [22]. On the other hand, even if the process has loops, the expanding of this process could be stopped by repeating a few times because the repeated information is enough for checking atomicity sphere. According to our investigation from the MIT process handbook [13], the number of states in most existing business processes ranges between 20 and 50, so it is practical to determine whether the well-formedness assumption is satisfied before applying our approach.

7. RELATED WORK

In this section, we will review some major techniques proposed by recent studies in the areas of transactional support, quality of services (QoS), and privacy protection for B2B collaboration.

To guarantee consistency in process collaboration, some researchers extend advanced transaction models (ATMs) [15] to support business processes. Grefen et al. [9] extends the Saga model [7] to facilitate an arbitrary distribution of and flexible rollback semantics of business processes over multiple workflow management systems. To keep consistency, each task in this model is required to be paired with a compensating task so that the failure of any task could be handled by compensating all or partial executed tasks. However, this requirement is too strict for B2B transaction because some tasks may not be able to compensate in reality. Similar works in the area of transactional workflow, such as [8], however compromise the autonomy and privacy of involved processes. In addition, some transactional protocols for process collaboration and web services, such as Business Transaction Protocol (BTP) [2], WS-C[19]/WS-T[20], use mechanisms like two-phase outcome protocol to guarantee the consistency of involved partners at runtime, which, however, sacrifices the autonomy of the participating processes because some involved processes may have to commit their tasks together.

Another well-studied area is distributed, cooperative fault-tolerant systems [11], where consistency is achieved by rolling the execution back to some checkpoints using backward recovery and restarting the execution from the checkpoints in case of a failure. These approaches, however, are inapplicable to processes in B2B environment because business processes may not be able to compensate executed tasks in some situations due to expensive costs. Moreover, tasks failed in a business process may not be able to restart. For example, the payment could not restart if the bank account is out of money. In addition, privacy issues are not considered in these approaches.

The work in this paper and our previous work [21] provide a flexible approach to achieve atomicity sphere property in B2B collaboration. Business processes do not have to conform to some specific protocols at runtime. As a result, the involved organizations have greater freedoms (i.e., loose coupling) to conduct B2B transactions. The main difference between the work in this paper and [21] is that the work of [21] uses a global checking approach, which may be inapplicable in some scenarios. This paper adopts a local analysis approach to analyze atomicity sphere when the global approach is inapplicable, and thus complements the work of [21].

Currently, little efforts have been spent on studying the privacy issue in B2B environment. Public view is a promising way to protect privacy, but existing works on public view based process collaboration, such as [3][4][12][17], could not be used to check atomicity sphere of B2B collaboration. Local views used in our work not only protect privacy of involved process, but also propagate non-compensable property so that business parties can analyze atomicity sphere locally without disclosing their privacy.

Lastly, we review related work in the area of quality of services (QoS) for process collaboration. Wohlstadter et al. [18] tackle the QoS problem in service matchmaking by using a middleware based methodology. This approach allows services to use a meta-level protocol to negotiate with one another and exchange their QoS policy information so that they can compute the agreeable common sets of QoS policies. Although their work may be used by the parties to negotiate transactional configurations, their work does not show the method to do so. We view that it is non-trivial, because atomicity sphere is different from the QoS aspects stated in [18] in the sense that atomicity sphere may be affected by all the involved processes. Other works on QoS aspect for service computing, such as [16][23], do not consider atomicity (sphere). In contrast, our work provides a viable approach for business parties to exchange their atomicity information and check locally whether atomicity sphere is held in their collaboration.

8. CONCLUSIONS

This paper presents a novel approach to analyze atomicity sphere locally for B2B collaboration without compromising privacy.

This method provides a criterion for business parties to identify suitable partners. It is based on a novel process algebraic model, under which different local views are derived from a private process for different roles. With the aid of these local views, property propagation is applied to localize any potential violation of atomicity sphere to individual private processes. By analyzing the business process using the proposed local atomicity sphere criterion at each involved process, business parties could identify whether their collaboration exhibits any atomicity sphere. In this way, each party discloses minimal privacy information to the others during the checking of atomicity sphere. Algorithms are also provided to construct the local views, propagate property and check the local criterion.

Our present work still has some limitations, e.g., trustworthiness is required for involved processes and the like. In the future, we will improve our work to address these restrictions and perform more case studies to investigate the usability and scalability of our approach in practice.

9. ACKNOWLEDGEMENTS

The authors would like to thank anonymous reviewers for their helpful comments and suggestions. Thanks are also given to Sam, Ivan, Rubin and other members of the software engineering group of HKUST for their informative discussions.

10. REFERENCES

[1] Brand, D., and Zafiropulo, P. On communicating finite-state machines. *Journal of the Association for Computing Machinery,* vol.30, no.2, Apr. 1983, pp. 323–342.

[2] BTP. Available at: http://www.oasis-open.org/committees/ tc_home.php?wg_abbrev=business-transaction (accessed on Apr 7, 2006).

[3] Bussler, C. Public process inheritance for business-to-business integration. In *Proceedings of 3rd International Workshop on Technologies for E-Services.* Springer-Verlag. 2002, pp.19–28. Berlin, Germany.

[4] Chiu, D.K.W., Cheung, S.C., Kafeza, E., and Leung, H.F. A three-tier view-based methodology for M-services adaptation. *IEEE Transactions on Systems, Man, & Cybernetics Part A: Systems & Humans*, vol.33, no.6, Nov. 2003, pp.725–741.

[5] CrossFlow Consortium / AGFIL. Insurance Requirements, *CrossFlow deliverable: D1.b.* Mar.1999. Available at: http://www.crossflow.org/public/pubdel/D1b.pdf (accessed on Mar 30, 2006)

[6] Dolev, D., Dwork, C., and Stockmeyer, L. On the minimal synchronism needed for distributed consensus. *Journal of the Association for Computing Machinery,* vol.34, no.1, Jan. 1987, pp. 77–97.

[7] Garcia-Molina, H., and Salem, K. SAGAS. *Sigmod Record*, vol.16, no.3, Dec. 1987, pp.249–259.

[8] Georgakopoulos, D., Hornick, M.F., and Manola, F. Customizing transaction models and mechanisms in a programmable environment supporting reliable workflow automation. *IEEE Transactions on Knowledge and Data Engineering,* vol.8, no.4, Aug. 1996, pp. 630-649.

[9] Grefen, P., Vonk, J., and Apers, P. Global transaction support for workflow management systems: from formal specification to practical implementation. *VLDB Journal*, vol.10, no.4, Dec. 2001, pp.316–333.

[10] Hagen, C., and Alonso, G. Exception handling in workflow management systems. *IEEE Transactions on Software Engineering*, vol.26, no.10, Oct. 2000, pp.943–958.

[11] Lee, P.A., and Anderson, T. *Fault tolerance: Principles and practice*, Springer-Verlag 1990.

[12] Liu, D.R., and Shen, M.X. Workflow modelling for virtual processes: an order-preserving process-view approach. *Information Systems,* vol.28, no.6, Sept. 2003, pp.505–532.

[13] MIT process handbook online, available at: http://process.mit.edu/Info/CaseLinks.asp (accessed on Apr 7, 2006)

[14] Schuldt, H., Alonso, G., Beeri, C., and Schek, H.J. Atomicity and isolation for transactional processes. *ACM Transactions on Database Systems,* vol.27, no.1, Mar. 2002, pp. 63–116.

[15] Sushil, J., and Larry, K. *Advanced Transaction Models and Architectures.* Kluwer 1997.

[16] Tai, S., Mikalsen, T., Wohlstadter, E., Desai, N., and Rouvellou, I. Transaction policies for service-oriented computing. *Data & Knowledge Engineering*, vol.51, issue 1, Oct. 2004, pp.59–79.

[17] van der Aalst, W.M.P., and Kumar, A. XML-based schema definition for support of interorganizational workflow. *Information Systems Research*, vol.14, no.1, Mar. 2003, pp.23–46.

[18] Wohlstadter, E., Tai, S., Mikalsen, T., Rouvellou, I., and Devanbu, P. GlueQoS: middleware to sweeten quality-of-service policy interactions. In *Proceedings of 26th International Conference on Software Engineering*. IEEE Comput. Soc. 2004, pp.189–199. Los Alamitos, CA, USA

[19] WS-C. Available at: http://msdn.microsoft.com/library/en-us/ dnglobspec/html/ws-coordination.asp (accessed on Apr 7, 2006).

[20] WS-T. Available at: http://msdn.microsoft.com/ws/2002/08 /wstx/ (accessed on Apr 7, 2006).

[21] Ye, C.Y., Cheung, S.C., and Chan, W.K. Publishing and Composition of Atomicity-equivalent Services for B2B Collaboration. In *Proceedings of the 28th International Conference on Software Engineering*, May. 2006, pp.351–360, Shanghai, China.

[22] Ye, C.Y., Cheung, S.C., and Chan, W.K. *on Local analysis of atomicity sphere for B2B collaboration*. Technical report, No.HKUST-CS-06-03, Department of Computer Science and Engineering, The Hong Kong University of Science & Technology, 2006.

[23] Zeng, L.Z., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., and Chang, H. QoS-aware middleware for Web services composition. *IEEE Transactions on Software Engineering,* vol.30, no.5, May. 2004, pp. 311–327.