# Detection Tests for Identifying Violators of Multi-party Contracts

Lai Xu
Utrecht University, The Netherland
Manfred A. Jeusfeld
Tilburg University, The Netherlands
Paul.W.P.J. Grefen
Eindhoven University of Technology, The Netherlands

In the modern business world, business coordinations are becoming complex and global. Contractual relations are required to explicitly define the collaboration between parties, certainly in scenarios with more than two parties. Therefore, execution of multi-party contracts has to be considered a vital point for successful business coordinations. However, there is little known about how to formally model a multi-party contract. The need for a multi-party contract model is thus becoming evident. In this paper, we investigate how to model a contract involving multilateral relations such that finding the responsible parties for given contract violations is facilitated. We provide algorithms for detecting violations and identifying the violator.

Categories and Subject Descriptors: H.4.0 [**Information Systems Applications**]: General

Additional Key Words and Phrases: E-contract, Contract violation, Detecting contract violator

## 1. INTRODUCTION

In the modern business world, we see that explicit collaboration between organizations is becoming more and more important. This is reflected in the emergence of tightly-coupled supply chains, the service outsourcing paradigm, complex comakerships, etcetera. Collaboration is not limited by geographical proximity, but increasingly of an international character. As a result of this development, explicit multi-party business coordinations are becoming global. As business collaboration is typically based on explicit contracts between collaborators, the execution of multi-party contracts has to be considered a vital point for successful business collaboration. In dynamic e-business settings, these contracts have to be of an electronic type that allows (semi)automatic handling to obtain the required levels of efficiency, speed and availability. The need for a multi-party contract model for an

Authors' address: Lai Xu (xu@cs.uu.nl), Institute of Information and Computing Sciences, Utrecht University, The Netherlands; Manfred A. Jeusfeld (manfred.jeusfeld@uvt.nl), CRISM/Infolab, Tilburg University, The Netherlands; Paul.W.P.J. Grefen (p.w.p.j.grefen@tm.tue.nl), Information Systems Group, Eindhoven University of Technology, The Netherlands.

e-business context is thus becoming evident [Xu 2004c], [Xu and Jeusfeld 2004].

In this paper, we present a multi-party contract model and provide how to detect responsible parties for a multi-party contract violation by using our model. Section 2 introduces our multi-party contract model. A detection method of a contract violation and an example for using this method are presented in Section 3. The paper ends with conclusions and a short discussion of further work in Section 4.

## 2. CONTRACT MODEL

A contract is an agreement between two or more parties that is binding to those parties and that is based on mutual commitments [Weigand and Xu 2001]. Our multi-party contract model consists of three core components: actions, commitments and a commitment graph [Xu and Jeusfeld 2003], [Xu 2003], [Xu 2004c]. A standard multi-party car insurance case [Project 1999] is used to explain our model. The case detailed description can be found from [Xu and Jeusfeld 2004].

### 2.1 Actions

Contractual parties perform actions as an imposed requirement which often have restrictions on the contract. An *action* describes what each partner should do. Moreover, the contract is explained by a set of commitments, a contractual party can thus be involved in different commitments and play the different roles, we specify the roles of a party as $\mathcal{R}$. The set of all roles of a contract is denoted as $\mathbb{R}$.

**Definition 1** *A party can act under different roles in different commitments. Let $ID$ be a domain of identifier;* **roles of a party** $\mathcal{R}_x$ *is defined as*

$$\mathcal{R}_x \subseteq ID.$$

*Let $P$ be a set of parties, the* **set of all roles** *is*

$$\mathbb{R} = \bigcup_{\forall x \in \mathcal{P}} \mathcal{R}_x.$$

**Definition 2** *Let $\mathbb{R}$ be a set of all roles of all parties, $ID$ be the domain ID, and $\mathbb{T}$ be the domain of time. An* **action** *is specified as*

$$action = (name, sender, receiver, deadline),$$

*where $name \in ID, sender, receiver \in \mathbb{R}$ and $deadline \in \mathbb{T}$. We require all names of actions to be unique so they can be used as identifiers.*

*A set of actions $\mathbb{A}$ for a contract can be specified as*

$$\mathbb{A} = \bigcup_{\forall x \in P} \{action\}.$$

For example, action (A_agreeRepairCar,L,G",3.5) describes that Lee C.S. agrees the garage to repair the car during the car damage claim received 3.5 days. For the car insurance case, all actions are specified in [Xu 2004a]. Actions will form the edges in commitment graphs. Although only a single receiver of the action are specified in the car insurance case, a list of action receivers can be extend in this model.

We have explained that different contractual parties play different roles. Each role has a set of pre-determined properties, whose values are part of the contract. To specify role properties in our contract model is a significant difference with

the multi-party contract model we presented in [Xu 2004b], [Xu 2004c], and [Xu and Jeusfeld 2004]. The role property consists of three parts which are inputs, outputs and rules of the role property. The inputs and outputs of role properties are domain related. The rules of the role properties are specified as the set of rules using predicate logic. The input of a role property determines the actions that a contractual party will take as a means to fulfill its obligations as specified in the contract. The output of the role property determines the objects which are the results of the executing action. When the party that implements a role attempts to execute an action, it first checks whether the input of the role property is satisfied, and subsequently generates the output of the role property. A formal definition of the role property is specified as

**Definition 3** *Let $\mathbb{R}$ be the set of all roles, $\mathbb{I}$ be the set of all information or objects involved in the contract. The rule is specified as*

$$rule : predicate1, predicate2, \cdots \rightarrow conclusion$$

*where predicate1, predicate2, $\cdots$, conclusion $\in \mathbb{I} \cup \mathbb{A}$.*

*The set of all rules $\mathbb{V}$ in the role properties is*

$$\mathbb{V} = \bigcup_{\forall role \in \mathbb{R}} \{rule\}.$$

*The role property is specified as*

$$property = (role, input, output, rules)$$

*where role $\in \mathbb{R}$; input, output $\in \mathbb{I}$; and rules $\in \mathbb{V}$.*

*A set of role properties $\mathbb{P}$ for a contract can be specified as*

$$\mathbb{P} = \bigcup_{\forall role \in \mathbb{R}} \{property\}.$$

Each role specifies values of input and output of the role property, e.g. in Table I, the third row, input element $ClaimForm$ indicates an empty claim form and input element $ClaimForm_{fi}$ illustrates a filled claim form with policyholder's data and signature. The rules of the role properties are specified as a set of rules using predicate logic. The conditions of the rules can be a conjection of the inputs of the role properties or a conjection of the inputs of the role properties and actions. For example, in the car insurance case, the garage ($G'$) plays a repairer role in the repair service commitment C_dailyService. The input of the role property for $G'$ is "$\text{Car}_{damaged}$" (i.e. $G'$ receiving a damaged car). According to the rule $Car_{damaged} \rightarrow$ A_estimateRepairCost, the garage $G'$ will fulfill the action A_estimateRepairCost after received the damaged car $\text{Car}_{damaged}$. According to the rule A_estimateRepairCost $\rightarrow estimatedRC$, the occurrence of A_estimateRepairCost will cause that the garage knows the estimated repair cost $estimatedRC$. All properties of the roles in the car insurance case are shown in Table I.

## 2.2 Commitments

In this paper, a commitment is a guarantee by one party towards another party that some action sequence shall be executed completely provided that some "trigger, involve, and finish" action happens, and all involved parties fulfill their side of the

| Role | Role Properties | | |
|---|---|---|---|
| | Input | Output | Rules |
| P' | assignedG, $Car_{fixed}$. | Claim, Records1, $Car_{damaged}$. | $PS_1 \rightarrow$ Claim;<br>$PS_2 \rightarrow$ Records1;<br>assignedG $\wedge$ $RS_1 \rightarrow Car_{damaged}$. |
| P" | ClaimForm | $ClaimForm_{fi}$ | ClaimForm $\rightarrow PR_2(CF_3)$;<br>$PR_2(CF_3) \rightarrow ClaimForm_{fi}$. |
| E | Records1 | assignedG, Records2 | Records1 $\rightarrow PS_3$;<br>$PS_3 \rightarrow$ assignedG;<br>Records1 $\wedge$ assignedG $\wedge$ $PS_4(CF_1, RS_1) \rightarrow$ Records2. |
| AG | Records2, $ClaimForm_{fi}$, Invoice. | Records3, ClaimForm, Payment. | Records2 $\rightarrow DS_2$;<br>Records2 $\rightarrow CF_2$;<br>$DS_2 \rightarrow$ Records3;<br>$CF_2 \rightarrow$ ClaimForm;<br>$ClaimForm_{fi} \wedge$ Invoice $\rightarrow PR_3$;<br>$PR_3 \rightarrow$ Payment. |
| L | Records3, estimatedRC, NewRC, Invoice. | assignedA, agreeRepair(RC), Invoice. | Records3 $\rightarrow DS_3$;<br>**if** estimatedRC $> 500$ **then**<br>{estimatedRC $\rightarrow DS_5(IC_1)$;<br>$DS_5(IC_1) \rightarrow$ assignedA;<br>NewRC $\rightarrow DS_7(RS_3)_{RC=NewRC}$;<br>$DS_7(RS_3)_{RC=NewRC} \rightarrow$ agreeRepair(NewRC)}<br>**if** estimatedRC $\leq 500$ **then**<br>{ estimatedRC $\leq 500 \rightarrow DS_7(RS_3)_{RC=estimatedRC}$;<br>$DS_7(RS_3)_{RC=estimatedRC} \rightarrow$ agreeRepair(estimatedRC)}<br>Invoice $\rightarrow DS_{10}(PR_1)$;<br>$DS_{10}(PR_1) \rightarrow$ Invoice. |
| A | assignedA | NewRC | assignedA $\rightarrow IC_2$;<br>$IC_3(DS_6) \rightarrow$ NewRC. |
| G' | $Car_{damaged}$, agreeRepair. | estimatedRC', $Car_{fixed}$ | $Car_{damaged} \rightarrow RS_2$;<br>$RS_2 \rightarrow$ estimatedRC';<br>agreeRepair(RC) $\rightarrow RS_4(DS_8)$;<br>$RS_4(DS_8) \rightarrow Car_{fixed}$. |
| G" | estimatedRC' | Invoice, estimatedRC | estimatedRC' $\wedge DS_3 \rightarrow DS_4$;<br>$DS_4 \rightarrow$ emtimatedRC<br>$DS_9 \rightarrow$Invoice . |
| G"' | Payment | | |

Table I: Roles and Role Properties

transaction. To finish a commitment, more than one party must finish relevant actions. A multi-party contract includes one or more commitments, a commitment includes some actions which could be performed by multi-parties. Those actions can trigger, involve, and finish the commitment.

**Definition 4** *Actions'* **attributes** $\mathcal{U}$ *can be specified as*

$$\mathcal{U} = \{tr, in, fi\}.$$

*Let* $ID$ *be the domain ID,* $P$ *be a set of parties,* $N = \{1, 2, 3, \ldots\}$, $\mathbb{A}$ *be a set of actions. A* **commitment** *is specified as*

$$commitment = (name, sender, receiver, n, \{(a_1, u_1), (a_2, u_2), \ldots, (a_n, u_n) : a_i \in \mathbb{A}, u_i \in \mathcal{U}\}).$$

*where name is an identifier,* $name \in ID$*; sender and receiver are the contract parties,* $sender, receiver \in P$*;* $n$ *denotes the total number of all actions involved,* $n \in N$*;* $a_1, a_2, \ldots, a_n$ *denotes all actions involved in the commitment and their attributes* $u_1, u_2, \ldots, u_n$ *. We require all names of commitments to be unique so that they can be used as identifiers.*

*A set of commitments* $\mathbb{M}$ *can be specified as*

$$\mathbb{M} = \bigcup_{\forall x \in P} \{commitment\}.$$

*Let* $a_i \in \mathbb{A}$ *and* $m \in \mathbb{M}$*, a sequence function* $f_{position} : \mathbb{A} \times \mathbb{M} \rightarrow N$*,*

$$f_{position}(a_i, m) = \begin{cases} i & iff\ i\ is\ the\ sequence\ number \\ & of\ action\ a_i\ in\ commitment\ m. \\ undef & otherwise. \end{cases}$$

$f_{position}(a_i, m)$ *denotes the position of action* $a_i$ *in the commitment* $m$*.*

For example, in commitment C_repairService, the garage will offer the repair service to the policyholder (see Table II the third row). After the policyholder sends his/her car to the garage (action A_sendCar has a trigger attribute), the garage estimates the repair cost (action A_estimateRepairCost has a involved attribute). After the garage receives an agreement from Lee C.S. about the repair cost (action A_agreeRepairCar has a trigger attribute), the garage repairs the car (action A_repairCar has a finish attribute). Commitment C_repairService is specified as

$$(\mathsf{C\_repairService}, \mathsf{G}, \mathsf{P}, \{(\mathsf{A\_sendCar}, tr), (\mathsf{A\_estimateRepairCost}, in), \\ (\mathsf{A\_agreeRepairCar}, tr), (\mathsf{A\_repairCar}, fi)\})$$

For the car insurance case, all commitments are specified in [Xu 2004a]. Table II provides actions, commitments and their abbreviation and labels in the car insurance case. The actions and commitments can be regard as a direct mapping from a paper contract to an e-contract.

| Commitment | Classification of Actions and Commitments | | | Labels |
|---|---|---|---|---|
| | Trigger | Involve | Finish | |
| C_phoneService (PS) | A_phoneClaim | | | PS.1 |
| | | A_receiveInfo | | PS.2 |
| | | A_assignGarage | | PS.3 |
| | | | A_notifyClaim | PS.4, CF.1, DS1 |
| C_repairService (RS) | A_sendCar | | | RS.1 |
| | | A_estimateRepairCost | | RS.2 |
| | A_agreeRepairCar | | | RS.3, DS.6 |
| | | | A_repairCar | RS.4, DS.7 |
| C_claimForm (CF) | A_notifyClaim | | | CF.1, PS.4 |
| | | A_sendClaimForm | | CF.2 |
| | | | A_returnClaimForm | CF.3, PR.2 |
| C_dailyService (DS) | A_notifyClaim | | | DS.1, PS.4, CF.1 |
| | | A_forwardClaim | | DS.2 |
| | | A_contactGarage | | DS.3 |
| | | A_sendRepairCost | | DS.4 |
| | | A_assignAssessor | | DS.5, IC.1 |
| | | A_sendNewRepairCost | | DS.6, IC.3 |
| | | | A_agreeRepairCar | DS.7, RS.3 |
| | A_repairCar | | | DS.8, RS.4 |
| | | A_sendInvoices | | DS.9 |
| | | | A_forwardInvoices | DS.10, PR.1 |
| C_inspectCar (IC) | A_assignAssessor | | | IC.1,DS.4 |
| | | A_inspectCar | | IC.2 |
| | | | A_sendNewRepairCost | IC.3,DS.5 |
| C_payRepairCost (PR) | A_forwardInvoices | | | PR.1, DS.10 |
| | A_returnClaimForm | | | PR.2, CF.3 |
| | | | A_payRepairCost | PR.3 |

Table II: Commitments, actions and action abbreviations

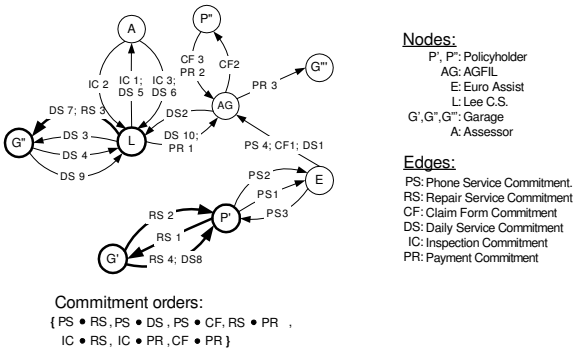


Fig. 1: Highlight repair service commitment C_repairService

## 2.3 Commitment Graph

Commitments are an even more important concept, though, to specify multi-party contracts. A commitment graph shows complex relationships among commitments. Commitment relationships are not only about a condition [Venkatraman and Singh 1999] or a chain relationship [Verdicchio and Colombetti 2002], [R.Ervin 2002]. For example, if a contractee first ships goods to a contractor, the contractor will pay the cost of goods later; the commitment of shipping goods is a condition to activate a commitment of payment.

Figure 1 shows the commitment graph for the car insurance case. All abbreviations and labels used in this commitment graph is provided in Table II. For all notes of this commitment graph, we use the following abbreviations: $P'$ and $P''$ for a policyholder, $AG$ for AGFIL, $E$ for Euro Assist, $L$ for Lee C.S., $G', G''$ and $G'''$ for garage, and $A$ for assessor. Each note represents a role that can be played by a contractual partner.

Each edge represents an action. Each action has one or more labels, where the first letter represents which commitments this action actually involves, the second number represents the order of a sequence actions within a commitment.

A commitment graph is a directed graph consisting of a set of nodes corresponding to all roles $\mathbb{R}$, a set of edges corresponding to actions and their labels, and commitment orders.

**Definition 5** *Let $\mathbb{A}$ be a set of actions, $a \in \mathbb{A}$, $\mathbb{M}$ be a set of commitments, $m \in \mathbb{M}$, and $X = \{1, 2, \ldots\}$, a sequence function $f_{position}(a, m)$, an edge is specified as a relation from $\mathbb{A} \times \mathbb{M} \times X$*

$$edge = \bigcup_{\forall m \in \mathbb{M}, a \in \mathbb{A}} \{(a, m, f_{position}(a, m)) : a \in \mathbb{A}, m \in \mathbb{M}, f_{position}(a, m) \in X\},$$

*a set of all edges is*

$$\mathbb{E} = \bigcup_{\forall a \in \mathbb{A}} \{edge\}.$$

**Definition 6** *Let $\mathbb{M}$ be a set of commitments. A commitment occurrence order is specified as a relation from $\mathbb{M} \times \mathbb{M}$:*

$$order\_commitment = \{(m_1 \cdot m_2) : m_1, m_2 \in \mathbb{M}, m_1 \neq m_2\}.$$

*If $m_1 \cdot m_2$ is a commitment order, we interpret it as follows: commitment $m_2$ is only active when commitment $m_1$ has been finished.*

*Let $P$ be a set of parties. A set of commitment orders lists all relationships in which a commitment occurs prior to another commitment, and is specified as follows:*

$$\mathbb{O} = \bigcup_{\forall x \in P} \{(m_1 \cdot m_2)\}.$$

For the car insurance case, examples of the commitment orders are presented in [Xu 2004a]. After specification of commitment graph notes, edges, and commitment occurrence orders, the commitment graph can be specified as follows:

**Definition 7** *Let $\mathbb{R}$ be a set of nodes, $\mathbb{E}$ be a set of edges, and $\mathbb{O}$ be a set of commitment order list. The commitment graph is defined as follows*

$$G = (\mathbb{R}, \mathbb{E}, \mathbb{O}).$$

### 2.4 Multi-party Contract Model

Now that all elements of our multi-party contract model have been presented, a formal model is provided as follows:

**Definition 8** *Let $\mathbb{A}$ be a set of actions, $\mathbb{M}$ be a set of commitments and $G$ be a commitment graph of a contract. The multi-party contract is specified as*

$$Contract = \{\mathbb{A}, \mathbb{M}, G\}.$$

## 3. DETECTING CONTRACT VIOLATORS

In the contract execution stage, figuring out the contract violators are the most important monitoring tasks. Papers [Xu 2004c] and [Xu and Jeusfeld 2004] discuss some special issues concerning contract violations in a multi-party relationship. The most commonly used detection process is to retrieve all actions that should have already occurred. Although it is a solution, this process is rather inefficient. Our approach uses a commitment graph and role properties to detect the parties responsible for the contract violation.

During the contract execution, a contractual party or a contract monitor finds out that a certain action cannot take after the other actions have been done. We call the action that cannot perform properly as $a_{miss}$, the action has been done as $a_{done}$. According to $a_{miss}$, $a_{done}$, and their positions in the commitment graph, the commitment graph can be restructured (or simplified). The process of restructuring the commitment graph is presented in Algorithm 1

First, finding all actions after action $a_{miss}$. Let us start with action $a_{miss}$, finding $a_{miss}$ involves in which commitment(s) $M_{current}$. Within the commitment ($m \in M_{current}$), the positions of actions ($\exists a', f_{position}(a', m)$) are higher than the position of $a_{miss}$, those actions do not yet occur and are put into $E_{not\_occur}$, if those actions are also involved in other commitments and there exists other actions which have higher positions, the new actions will be included in $E_{not\_occur}$. Besides, according to the commitment order, the commitments after $a_{miss}$'s commitment(s) are put into $M_{not\_trigger}$. All actions in commitments $M_{not\_trigger}$ are also sent to $E_{not\_occur}$.

All actions before action $a_{done}$ are finished. Therefore, looking at action $a_{done}$, locating $a_{done}$ involved in which commitment(s) $M_{current}$. Within the commitment

$m \in M_{current}$, the positions of actions $(\exists a', f_{position}(a', m))$ are before the position of $a_{done}$, those actions have been finished and are put into $E_{done}$. If those actions are also involved in other commitments and there exists other actions of the commitment which have lower positions, the new actions will be included in $E_{done}$ as well.

Finally, the set of all actions that need to be cut is the conjunction of $E_{not\_occur}$ and $E_{done}$. Function RestructureCG($E$) has the function to cut all edges in set $E$. If there are disconnected nodes, those nodes will also be cut away.

Based on the restructured commitment graph, we start to check $a_{miss}$, whether the inputs of the role properties have been received by the sender of $a_{miss}$. The process of detecting responsible partners of a contract violation has the following steps. The contractual party, who is playing a particular role, check this violation from the role property's input. If the violation is located at the role property's input, the outputs of other role properties, which have the same name as the input of the role property, need to be found. The action which actually causes this output of the role property need to be checked. If the action does not occur and the condition of the action occurrence is true, the sender of the action is the responsible party. Using this input continue to follow the above steps until the known facts have met. The detecting responsible parties of the contract violation is presented in Algorithm 2.We use a typical scenario to explain our algorithms.

In the car insurance case, after Lee C.S. contacted the garage, the garage did not send the estimated repair cost to Lee C.S. The scenario is a situation for two mutual depending commitments. One commitment cannot go on because another commitment does not start, or does not perform the action which should be performed. The two commitment share a contractual party which plays different roles in two different commitments. Thus, the role properties are used in detecting this sort of violators. Actually, the algorithm we presented in [Xu 2004c] presented does not provide an efficient process for this kind of the detection pattern. This also motivates why we extended our old contract model from [Xu 2003], [Xu 2004b] and [Xu and Jeusfeld 2003], into a new contract model in paper [Xu and Jeusfeld 2004].

In this scenario, $a\_miss$ =A_sendRepairCost $(DS_4)$, which should be performed by role $G'$ and $a\_done$ = A_contactGarage $(DS_3)$, which is finished by the role $L$.

**Input:** Contract = $\{\mathbb{A}, \mathbb{M}, G\}$
$action\_of\_miss : a_{miss}$ = A_sendRepairCost $(DS_4)$
$action\_of\_done : a_{done}$ = A_contactGarage $(DS_3)$
/* First, detecting all actions which have not occurred. */
$M_{current} = \{DS : edge(\text{A\_sendRepairCost}, DS, 4)\}$
$M_{not\_triggered} = \emptyset$
For action A_assignAssessor $((DS_5, IC_1)$, which means commitment C_inspectCar (IC) has not yet been triggered, therefore put $IC_2$ and $IC_3$ into $E_{not\_occur}$
For action A_agreeRepairCar $(DS_7, RS_3)$, therefore put $RS_4$ into $E_{not\_occur}$
For action A_forwardInvoices $(DS_{10}, PR_1)$, which means commitment C_payRepairCost (PR) has not yet been triggered, therefore put $PR_2$ and $PR_3$ into $E_{not\_occur}$
$E_{not\_occur} = \{DS_4, DS_5/IC_1, DS_6/IC_3, DS_7/RS_3,$
$DS_8/RS_4, DS_9, DS_{10}/PR_1, IC_2, PR_2, PR_3\}$
/* Second, getting all actions which have been finished.*/
$E_{done} = \emptyset$
$E_{cut} = \{DS_4, DS_5/IC_1, DS_6/IC_3, DS_7/RS_3,$
$DS_8/RS_4, DS_9, DS_{10}/PR_1, IC_2, PR_2, PR_3\}$

After invoking **Algorithm** 1, the new detecting graph is presented in Figure 2.

Starting with role $G$" (the garage), first of all, the garage will check whether

**Algorithm 1** Simplifying the commitment graph

**Input:** $Contract = \{\mathbb{A}, \mathbb{M}, G\}$
$action\_of\_miss : a_{miss}$
$action\_of\_done : a_{done}$
**Initialization:** $M_{not\_triggered} = \emptyset$
$E_{not\_occur} = \emptyset$
$M_{finished} = \emptyset$
$E_{done} = \emptyset$
$E_{cut} = \emptyset$
/* First, detecting actions which have not occurred. */

$$M_{current} = \bigcup_{\forall m, m \in \mathbb{M}} \{m, \exists edge, edge(a_{miss}, m, f)\}$$

$$M_{not\_triggered} = \bigcup_{\forall m, m \in M_{current}} \{m' : m.m' \in \mathbb{O}\}$$

**for** $m \in M_{current}$ **do**
  **for** $\exists a', f_{position}(a', m) > f$ **do**
    $E_{not\_occur} = \{a'\} \cup E_{not\_occur}$
    **for** $\exists m', \forall a'' : edge(a', m', f') \vee f_{position}(a'', m') > f \vee a'' \notin E_{not\_occur}$ **do**
      $E_{not\_occur} = \{a''\} \cup E_{not\_occur}$
    **end for**
  **end for**
**end for**
**for** $m \in M_{not\_triggered}$ **do**
  **for** $\exists a, m.action = a \vee a \notin E_{not\_occur}$ **do**
    $E_{not\_occur} = \{a\} \cup E_{not\_occur}$
  **end for**
**end for**
/* Second, getting actions which have been finished.*/

$$M_{current} = \bigcup_{\forall m, m \in \mathbb{M}} \{m, \exists edge, edge(a_{done}, m, f)\}$$

**for** $m \in M_{current}$ **do**
  **for** $\exists a', f_{position}(a', m) < f \vee a' \notin E_{done}$ **do**
    $E_{done} = \{a'\} \cup E_{done}$
    **for** $\exists m', \forall a'' : edge(a', m', f') \vee f_{position}(a'', m') < f \vee a'' \notin E_{done}$ **do**
      $E_{done} = \{a''\} \cup E_{done}$
    **end for**
  **end for**
**end for**
/* Thrid, restructure the commitment graph */
$E_{cut} = E_{not\_occur} \cup E_{done}$
RestructureCG($E_{cut}$)

**Algorithm 2** Detecting violators

Algorithm 1
$E$ is a set of all edges of the restructured commitment graph
Call check($a_{miss}$)
check(a: an action)
**repeat**
  $E = E - \{a\}$
  **if** $\exists input, property(a.sender, input, \_, \_) \wedge$ received$(input) = True$ **then**
    return($a.sender$)
  **else if** $\exists output, rule : property(a'.sender, \_, output, rule) \vee output = input \vee \exists a', a' \in rule, condition \vee output \in rule.result$ **then**
    Call check($a'$)
  **end if**
**until** $E = \emptyset$

the damaged car has been received. If the answer is "Yes", then the garage is a violator. Otherwise, the policyholder needs to be checked, e.g. whether the Euro Assist has assigned a garage to him/her. If a garage has been assigned to the policyholder and the policyholder did not send the damaged car, then the

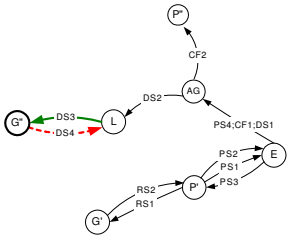


Fig. 2: Detection graph for the scenario

policyholder is a violator. Otherwise, Euro assist has to be checked whether Euro Assist gave consistent information to AGFIL, and whether AGFIL passed correct information to Lee C.S.

## 4. CONCLUSIONS

As main contribution this work addresses contract violation detection in the multi-party relations. To this respect a formal multi-party contract model and a formal method for violator detection are provided. The multi-party contract model as proposed consists of three parts: actions, commitments and the commitment graph. The violator detection method uses a commitment graph to trace back the commitments after a contract violation and locates the parties who violated the commitments.

In the web service world, contracts are becoming more important, certainly if web services are used for the enactment of complex business processes [P. Grefen and Angelov 2003]. In situations where web services link multiple collaborating parties, the work presented in this paper is an ingredient for a complete contracting solution.

## REFERENCES

P. Grefen, H. Ludwig, A. D. and Angelov, S. 2003. Web services support for dynamic business process outsourcing. *IBM Research Report RC22728; IBM Research Division*.

Project, C. 1999. Insurance requirements. Tech. Rep. CrossFlow deliverable: D1.b, CrossFlow consortium.

R.Ervin. 2002. Chains of commitment software architecture. *ACM SIGecom Exchanges 3,* 1.

Venkatraman, M. and Singh, M. P. 1999. Verifying compliance with commitment protocols: Enabling open web-based multiagent systems. *Autonomous Agents and Multi-Agent Systems 2,* 3.

Verdicchio, M. and Colombetti, M. 2002. Commitments for agent-based supply chain management. *ACM SIGecom Exchanges 3,* 1.

Weigand, H. and Xu, L. 2001. Contracts in e-commerce. In *9th IFIP 2.6 Working Conference on Database Semantic Issues in E-Commerce Systems (DS-9)*.

Xu, L. 2003. Monitorable electronic contract. In *The 2003 IEEE Conference on E-Commerce (CEC'03)*. IEEE Computer Society Press.

Xu, L. 2004a. Appendix: all actions and commitments of a car insurance case. `http://www.cs.vu.nl/~xu/appendix.pdf`.

Xu, L. 2004b. Monitoring multi-party contracts for e-business. Ph.D. thesis, Tilburg University.

Xu, L. 2004c. A multi-party contract model. *ACM SIGecom Exchanges 5,* 1.

Xu, L. and Jeusfeld, M. A. 2003. Pro-active monitoring of electronic contracts. In *The 15th Conference On Advanced Information Systems Engineering in Lecture Notes of Computer Science*. Vol. 2681. Springer-Verlag, 584–600.

Xu, L. and Jeusfeld, M. A. 2004. Detecting violators of multi-party contracts. In *The Conference on CoopIS/DOA/ODBASE in Lecture Notes of Computer Science*. Vol. 3290. Springer-Verlag, 526–543.