A π-calculus Formalization of Contract Violation Diagnosis

Gianluca Torta[1] and **Roberto Micalizio**[1]
[1]Università di Torino, Torino, Italy
e-mail: {torta,micalizio}@unito.it

Abstract

Many multiagent processes involving complex interactions among autonomous agents can be conveniently captured by so called Interaction Protocols (IPs), which specify rules for the creation of contracts (i.e., commitments) between the agents involved in the protocol execution. Unfortunately, the diagnosis of protocol violations is hindered by the fact that IPs do not specify why an agent may fail to comply with its commitments. In this paper, we define IPs as π-calculus processes, where the agents behavior is associated with the outcomes of decisions made by applying internal policies. While the agents keep their internal policies private, the public knowledge of relations between behaviors and decisions allows for a definition of diagnosis that is more explanatory than previous ones in the literature.

1 Introduction

Many multiagent processes involving complex interactions among autonomous agents (most notably, business processes that involve multiple business actors) can be conveniently captured by so called Interaction Protocols (IPs) [Desai *et al.*, 2005]. Unlike the classic workflow-based models [Van Der Aalst and Van Hee, 2004], such protocols are specified at a more abstract and distributed level, in terms of rules for the creation (and possibly other manipulations) of *commitments* [Castelfranchi, 1995; Singh, 1999] between the agents involved in the execution of an instance of the protocol. Each commitment can be thought of as a contract, and specifies a condition that an agent A (debtor) commits to bring about for an other agent A' (creditor); e.g., a customer commits to pay for an item when such an item is delivered.

While, given suitable observations, it can be relatively straightforward to monitor the status of commitments during the protocol execution, and detect commitment violations [Robinson and Purao, 2009; Kafalı and Yolum, 2009], things get more complicated when it comes to derive some kind of explanation (i.e., diagnosis) for such violations. The main obstacle lies in the fact that IPs only specify when agents commit with each other, and what such commitments are; but they do not capture why an agent may fail to comply with (one or more of) its commitments. Some researchers [Kafalı and Torroni, 2012] have proposed to explain the violation of a commitment in terms of a misalignment in the representation of the commitment between the debtor and the creditor. While this can produce interesting information about the failures due to misalignments, it fails to explain commitment violations when the debtor and creditor have identical beliefs about the commitment(s) that exist between them [Chopra and Singh, 2009].

In this paper, we adopt a representation of the IPs closely inspired by that of [Desai *et al.*, 2005], where, although the commitments existing between any two agents and the messages exchanged among them are publicly observable, the autonomy of the agents is ensured by keeping their internal *policies* private [1]. In particular, we define IPs as π-calculus processes, where the agents behavior is associated with the outcomes of decisions made by applying internal policies. The public knowledge of relations between behaviors and decisions allows for a definition of diagnosis that takes into account the decisions made by agents in explaining commitment violations. This yields explanations significantly more informative than the mere detection of violations provided by monitoring.

The paper is structured as follows. In section 2 we give some background on Interaction Protocols and the π-calculus, used in the paper to represent IPs. Section 3 presents a motivating example that will be used throughout the paper to illustrate our approach. Section 4 describes in detail our representation of IPs, including agents and commitments. Section 5 introduces formal definitions of Diagnostic Problem and Diagnosis in the context of IPs, and section 6 presents our approach to the computation of such diagnoses with the help of Model-Checking. Finally, section 7 reviews some related work and illustrates future research directions.

2 Background

2.1 Interaction Protocols

An Interaction Protocol (IP) describes a pattern of behavior that an agent has to follow to engage in a communicative interaction with other agents within a multiagent system [Fornara and Colombetti, 2003]. In this paper we adopt a commitment-based interaction protocol; such a protocol specifies a set of roles, the set of messages they can exchange, and the meanings of such messages expressed in terms of their effects on the commitments among the agents playing the specified roles.

[1]In [Desai *et al.*, 2005] the authors propose to describe and possibly share on the Web IPs through OWL-P, a OWL ontology for protocols.

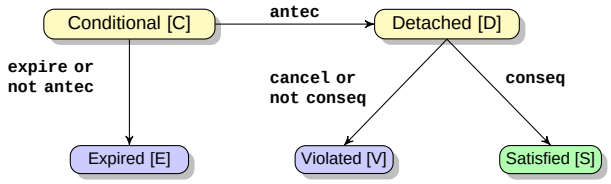

Figure 1: A simplified commitment life cycle.

A commitment is represented as:

$$cc(x, y, p, q)$$

where the debtor agent x commits towards the creditor y to bring about the consequent condition q when the antecedent condition p holds. In this paper, we shall assume that antecedent and consequent conditions are unary predicates $p(.)$, $q(.)$.

A commitment has a state that evolves over time according to the events that occur in the system. The commitment life cycle is formalized in [Telang *et al.*, 2011]; in this paper we adopt the simplified life cycle showed in Figure 1.

A commitment is `conditional` when it is created by its debtor agent. In such a state the debtor agent is not required to bring about the consequent condition since the antecedent has not occurred yet. In principle, however, the debtor could produce the consequent, and hence satisfy the commitment, even though the antecedent is missing. Since antecedent and consequent conditions could be associated with temporal constraints, as exemplified in [Kafalı and Torroni, 2012], a commitment can also `expire`; specifically, this happens when the antecedent condition does not occur before a predefined deadline. On the contrary, when the antecedent occurs a conditional commitment becomes `detached`, or `active`. This means that the debtor is now obliged to bring about the consequent lest being sanctioned by the creditor. A detached commitment can also be rewritten as a *base* commitment

$$cc(x, y, \top, q) \equiv c(x, y, q)$$

In case the debtor fails in producing the consequent, the commitment evolves into the `violated` state; the commitment is `satisfied`, otherwise. In this first proposal, we do not consider temporal constraints on antecedent and consequent conditions. Moreover, following [Kafalı and Torroni, 2012], we assume that a *coupled* Knowledge Base $KB_{i,j}$ exists between each pair of agents A_i, A_j that interact in the IP. These knowledge bases assure that both agents involved in any commitment are capable of inferring the existence and the state of the commitment binding them, by observing the occurrence of relevant events for the creation and satisfaction/violation of the commitment. On the other hand, we do not assume that the events relevant for a commitment are also observable by agents which are not involved in that commitment.

2.2 π-calculus

As we shall see, in this paper we model Interaction Protocols as π-calculus processes, following an idea proposed in [Desai *et al.*, 2005]. In the π-calculus [Milner, 1999], the basic entities are concurrent, communicating processes. Such processes can evolve by performing *actions*, namely sending and receiving *names* over *channels*. Figure 2 summarizes the syntax of π-calculus.

$\overline{a}(x)$	send action
$a(x)$	receive action
$\alpha.P$	prefix action (α is $\overline{a}(x)$ or $a(x)$)
$(new\ x)P$	local name
$P_1 + P_2$	non-deterministic choice
$P_1 \vert P_2$	parallel composition
$[x = y]P$	conditional execution (match)
$P(\ldots) \stackrel{def}{=} \ldots$	process definition
$P\langle \ldots \rangle$	process run

Figure 2: Basic syntax of the π-calculus.

Let us briefly review such a syntax by putting it at work within our context. In particular, consider how (a part of) a *Seller* agent taking part in a Interaction Protocol could be represented in π-calculus (adapted from [Desai *et al.*, 2005]):

$$\begin{aligned} &Seller(rfq, quote, accept, reject) \stackrel{def}{=} (new\ quote\_pol) \\ &\ rfq(g). \\ &\ (\overline{Quote_p}\langle quote\_pol\rangle \vert \\ &\ \ (\overline{quote\_pol}(g).quote\_pol(g, p). \\ &\ \ \ \widehat{quote}(g, p).(accept(g', p') + reject(g', p')))) \end{aligned}$$

The parameters *rfq*, *quote*, *accept*, and *reject* of the process definition are the channels through which the *Seller* will communicate with other agents (e.g., *Buyer*); while the local channel *quote_pol* is used by *Seller* to communicate with its own internal quoting policy, that given an item g decides its price p. Note that the send action on the *quote* channel is represented as $\widehat{quote}$ instead of $\overline{quote}$, meaning that the message is broadcast to all processes reading on the *quote* channel. This is needed for tracking the state of commitments, as we shall see below.

The process starts by receiving the item g to quote from channel *rfq*; it then activates its quoting policy process $Quote_p$ (passing the local channel needed for communication as a parameter), in parallel with a sequence of other actions. Such a sequence starts by sending g on the *quote_pol* channel, and receiving the price p returned by $Quote_p$ from the same channel. Then, the price p is sent to all the processes reading on the *quote* channel, including the *Buyer*. Finally, the *Seller* waits for the decision of the *Buyer* to arrive from either the *accept* or *reject* channel.

Similarly, the corresponding part of the *Buyer* may be defined as follows:

$$\begin{aligned} &Buyer(rfq, quote, accept, reject) \stackrel{def}{=} (new\ rfq\_pol, acc\_pol) \\ &\ (RFQ_p\langle rfq\_pol\rangle \vert \\ &\ \ (\overline{rfq\_pol}.rfq\_pol(g).\widehat{rfq}(g).quote(g', p'). \\ &\ \ \ (\overline{Acc_p}\langle acc\_pol\rangle \vert \\ &\ \ \ \ (\overline{acc\_pol}(g', p').acc\_pol(g'', p'', res). \\ &\ \ \ \ \ ([res = \texttt{yes}]\widehat{accept}(g'', p'') + [res = \texttt{no}]\widehat{reject}(g'', p'')))))) \end{aligned}$$

As above, *rfq*, *quote*, *accept*, and *reject* are channels for exchanging messages between agents. The agent activates its request policy RFQ_p (which determines when the Buyer will need a quote, and for what item g) and, after receiving the quote price p' from the *Seller*, it activates its accept/reject policy Acc_p. The *Buyer* agent may take two different paths based on the decision *res* returned by Acc_p on the *acc_pol* channel; such alternatives are guarded by suitable *match* conditions on the value of *res*.

3 Motivating Example

Let us consider a simple example on how this can help diagnosing commitment violations. Consider a *Bookstore* agent (inspired from [Kafalı and Yolum, 2009]), with the following (simplified) π-calculus definition[2]:

$$Bookstore(pay, orderdel, deliver) \stackrel{def}{=} pay(bk).ProcessOrder\langle orderdel, deliver, bk, \texttt{1st}\rangle$$

$$ProcessOrder(orderdel, deliver, bk, cnt) \stackrel{def}{=} (new\ po\_pol)\ (ProcOrder_p\langle po\_pol\rangle | \overline{(po\_pol}(bk).po\_pol(bk, res).([res = \texttt{ready}]\ \widehat{orderdel}(bk) + [res = \texttt{wait}][cnt = \texttt{1st}]\ ProcessOrder\langle orderdel, deliver, bk, \texttt{2nd}\rangle + [res = \texttt{wait}][cnt = \texttt{2nd}]\ \widehat{deliver}(\texttt{no}))))$$

The *Bookstore BS* waits for a book payment to arrive on the *pay* channel, and then goes on to process the order (call to *ProcessOrder*). The process calls the local policy process $ProcOrder_p$ and, in parallel, it queries such process about *bk* through the *po_pol* channel, receiving a *res*ult: either a large enough batch of orders have been received (*ready*) or not yet (*wait*). In the former case, *ProcessOrder* sends the order to the *Shipper SH* through the *orderdel* channel. Otherwise, it tries one more time (with *cnt* bound to `2nd`) and, if the policy asks to wait again, it fails (sending `no` on the *deliver* channel). Note that it is important to explicitly associate a value of *res* to each alternative path, in order to be able to track the decision outcome that causes the *Bookstore* agent to choose one path over another one.

The interaction protocol between *Customer* (CS) and *BS* contains the following conditional commitment:

$$c_{DEL} = cc(BS, CS, pay(\mathbf{g}), deliver(\mathbf{g}))$$

stating that, if *CS* pays a good **g**, then *BS* guarantees that **g** is delivered (eventually). Such a commitment becomes active when *BS* actually receives a name (e.g., `hp7` for "Harry Potter 7") on the *pay* channel:

$$cc(BS, CS, \top, deliver(\texttt{hp7}))$$

Let us assume that, at some later time, the commitment is violated, i.e. a message *deliver(*`no`*)* is generated by some of the agents. In such a case, *CS* may ask *BS* for what happened. In turn, *BS* considers the trace of labels of its behavior:

$$\underline{pay}(\texttt{hp7}).\overline{po\_pol}(\texttt{hp7}).po\_pol(\texttt{hp7}, \texttt{wait}).$$
$$\overline{po\_pol}(\texttt{hp7}).po\_pol(\texttt{hp7}, \texttt{wait}).\widehat{deliver}(\texttt{no})$$

figuring out that its local policy has asked twice to *wait* for more book orders before contacting *DEL* for actual delivery. It also realizes that, if it got the alternative reply `ready` at least once, there would have been a path to satisfy the consequent *deliver(*`hp7`*)*. The sequence of internal decisions that caused the violation is therefore:

$$po\_pol(\texttt{hp7}, \texttt{wait}).po\_pol(\texttt{hp7}, \texttt{wait})$$

If it wishes so, *BS* can return to *CS* such a sequence (or some message derived from it), which explains the violation with the fact that it was waiting for more orders.

[2] In the original example, the commitments had *deadlines*; currently, we only deal with atemporal commitments.

4 Modeling Interaction Protocols

As said above, we use π-calculus to model Interaction Protocols. Let us start by defining how a role participating in the interaction protocol is modeled. Following [Desai *et al.*, 2005], we name such a representation Role Skeleton.

Definition 1 *A Role Skeleton is a π-process:*

$$Role(ch_1, \ldots, ch_p) \stackrel{def}{=} (new\ pch_1, \ldots, pch_q)P_{Role}$$

where $ch_1, \ldots, ch_p$ are the (global) channels through which Role *communicates with other agents, while $pch_1, \ldots, pch_q$ are the (local) channels through which* Role *communicates with its internal policies.*
We require that the interactions of the agent with its internal (local) policies follow this scheme:

$$(LP\langle pch\rangle | (\overline{pch}(x_1, \ldots, x_k).pch(x_1, \ldots, x_k, y).P_{cont}))$$

where LP is the process that implements the local policy, which echoes back on the pch channel the input parameters $x_1, \ldots, x_k$ received from the Role*, and returns a result in the y variable.*
Moreover, we require that the Role *explicitly models through the* match *construct the influence of the internal policy result on the choice among alternative continuations of its process, i.e., P_{cont} has the following form:*

$$[y = v_1]P_1 + \ldots + [y = v_m]P_m$$

We assume the adoption of the *match* construct to make explicit the causal relation between internal policies and agent's behavior, so as to facilitate diagnosis.

We will denote the Agent process corresponding to an agent A_i as $\mathcal{A}_i$; such a process is simply an instantiation $\mathcal{R}\langle ch_1, \ldots, ch_p\rangle$ of a Role Skeleton $\mathcal{R}$. While in general more than one agent may play the same role within a given IP, we shall assume for simplicity that each role is played by exactly one agent.

Also commitments can be modeled as π-calculus processes. Compared to [Desai *et al.*, 2005], we want to explicitly model the state transitions of each commitment, in order to facilitate diagnosis. We assume that the life cycle of the commitment can touch states `cond` (conditional), `act` (active), `sat` (satisfied), and `viol` (violated) (see Figure 1). A commitment is then modeled as the folowing π-calculus process:

$$CC(id, deb, cred, ant, aok, cons, cok) \stackrel{def}{=} (new\ aval, cval, st).id(st).((cons(cval).([cval \neq cok](CC\langle \ldots\rangle | \widehat{id}(\texttt{viol})) + [cval = cok](CC\langle \ldots\rangle | \widehat{id}(\texttt{sat})))) + (ant(aval).([aval \neq aok](CC\langle \ldots\rangle | \widehat{id}(\texttt{cond})) + [aval = aok](CC\langle \ldots\rangle | \widehat{id}(\texttt{act})))))$$

where:

- *id* is the unique name of the commitment
- *deb*, *cred* are the debtor and creditor of *id*
- *ant* is the unary predicate of the antecedent condition of *id*
- *aok* is the value that satisfies the antecedent condition
- *cons* is the unary predicate of the consequent condition of *id*
- *cok* is the value that satisfies the consequent condition

The process starts by receiving from *id* the status of the commitment. Then, it either receives a *cval* for *cons* or an *aval* for *ant*. In case it receives a value *cval* for *cons* the process calls itself recursively transitioning to state `sat` if *cval* is equal to the *cok* value, and to state `viol` otherwise. In case, on the other hand, the process receives a value *aval* for *ant*, it transitions to state `act` if *aval* is equal to *aok*, and back to state `cond` otherwise.

Let us now define the coupled Knowledge-Bases as π-calculus processes.

Definition 2 *A KB Skeleton is a π-process:*

$$KB(r1, r2, ch_1, \ldots, ch_p) = (PR_1 | \ldots | PR_n)$$

$$PR_i = ch(\vec{v}).(CC\langle id_i, r_j, r_k, \ldots\rangle | \widehat{id_i}(\texttt{cond}))$$

where:

- $ch_1, \ldots, ch_p$ *are the (global) channels through which* KB *communicates with the roles* $r1$, $r2$
- *each* PR_i *is a Protocol Rule that, when a channel* ch *receives an array of values* $\vec{v}$*, creates a new commitment* id_i *in status* `cond` *with debtor* $r1$ *and creditor* $r2$ *or vice versa.*

Example 1 *Let us consider the commitment* c_{DEL} *introduced above in the Bookstore scenario, and see how the commitment is created by the KB binding the bookstore BS and the customer CS:*

$$(CC\langle c_{DEL}, BS, CS, pay, \texttt{hp7}, deliver, \texttt{hp7}\rangle | \widehat{c_{DEL}}(\texttt{cond}))$$

The KB, without requiring the arrival of any specific message $ch(\vec{v})$*, immediately creates a process* CC *whose name,* c_{DEL}*, is indeed a channel used to trace the state of the commitment itself. At the beginning, the commitment is conditional and this state is broadcast to all entities interested in knowing it (with action* $\widehat{c_{DEL}}(\texttt{cond})$*). The KB passes to the new commitment process two channels* pay *and* deliver*, through which the commitment process can observe the behavior of bookstore and customer. Specifically,* pay *captures the "paying activities" of the customer, whereas* deliver *captures the delivery of the required item to the customer. The KB also passes two values, namely the argument that makes the antecedent condition satisfied in order to make the commitment evolve into* `active`*, and the argument that the consequent must take to consider the commitment as satisfied. In our simple case, since the customer pays for* `hp7` *and since she wants it delivered, both values are* `hp7`.

Let us now suppose that the customer actually pays for the requested item. Such an event is observed by all the processes waiting on channel pay*; namely, the bookstore, and the commitment* c_{DEL}*. Within the commitment process* c_{DEL}*, the reception of this event initializes value* aval *as* `hp7`*, and consequently the commitment process evolves along the following execution path:*

$$[aval = \texttt{hp7}](CC\langle\ldots\rangle | \widehat{c_{DEL}}(\texttt{act}))$$

which actually consists of a recursive call of the commitment process where, this time, the commitment state is evolved to `active`*.*

Assume now that the for some reasons the delivery of the requested item does not occur; that is, we have evidence that the delivery will never happen. This is captured by receiving a message on channel deliver*, which initializes value* cval *with the default value* `no`*. The commitment process takes the path:*

$$[cval \neq \texttt{hp7}](CC\langle\ldots\rangle | \widehat{c_{DEL}}(\texttt{viol}))$$

meaning that the commitment state has now evolved into `violated`*.*

An Interaction Protocol is simply given by the parallel composition of a set of instances of Role Skeletons and KB Skeletons.

Definition 3 *A multi-agent Interaction Protocol (IP) is a π-process:*

$$\mathcal{IP} = (\mathcal{A}_1 | \ldots | \mathcal{A}_n) | \{\mathcal{KB}_{i,j}\}$$

where each agent process $\mathcal{A}_i$ *is an instance of a Role Skeleton participating in the protocol and each KB process* $\mathcal{KB}_{i,j}$ *is an instance of a KB Skeleton.*

5 Diagnostic Problems and Diagnosis

A Diagnostic Problem is posed to an agent A_i when a commitment involving A_i as debtor is violated.

Definition 4 *A Diagnostic Problem for agent* A_i *is a tuple:*

$$\mathcal{DP}(A_i) = (\mathcal{IP}, cc(A_i, A_j, p, q), \sigma_i, \{\sigma_{i,j}\})$$

where:

- $\mathcal{IP}$ *is a IP process*
- $cid = cc(A_i, A_j, p, q)$ *is a violated commitment with debtor* A_i
- σ_i *is the trace of all the actions performed by process* $\mathcal{A}_i$ *so far*
- *each* $\sigma_{i,j}$ *is the trace of all the actions performed by process* $\mathcal{KB}_{i,j}$ *so far*

The system is represented by an Interaction Protocol process $\mathcal{IP}$, while the available observations include both the trace of actions σ_i performed by A_i and the actions $\sigma_{i,j}$ capturing the creation and evolution of commitments between A_i and other agents A_j. We assume that observations in σ_i and $\sigma_{i,j}$ are timestamped, so that we can define a merged trace:

$$\sigma_{DP} = a_1 \ldots a_n$$

where the a_is are the actions appearing in traces σ_i and $\sigma_{i,j}$s, ordered by their timestamps (with ties arbitrarily resolved).

Comparing the present definition with the classic Model-Based definition of a Diagnostic Problem [Reiter, 1987]:

$$(SD, COMPS, OBS)$$

it is natural to map SD (System Description) to $\mathcal{IP}$, and OBS (Observations) to trace σ_{DP}. It is somewhat less obvious to find a counterpart for the assumables $COMPS$, and more generally for the Diagnoses that, in classic MBD, are just minimal subsets Δ of $COMPS$ s.t. the assumption of the abnormality $AB(c)$ for each $c \in \Delta$ is consistent with SD and OBS. In our context, a (Candidate) Diagnosis consists of a sequence of: choices made by A_i relying on its local policies, values received by A_i, and commitments violated by other agents.

Definition 5 *A Candidate Diagnosis for problem* $\mathcal{DP}(A_i)$ *is a sequence:*

$$\Delta = \delta_1, \ldots, \delta_m$$

s.t. Δ *is a projection of* σ_{DP} *where each* δ_j *is either:*

- *a receive action* $lp(\vec{in}, out)$ *where a local policy has returned* out
- *a receive action* $ch(\vec{v})$ *where some agent has sent* $\vec{v}$
- *a violated commitment* $cid'(\texttt{viol})$ *whose creditor is* A_i

In order to be a (correct) Diagnosis, a CD Δ must entail the failure of commitment cid, and it must be minimal (i.e., no prefix of Δ entails the failure of commitment cid).

Definition 6 *A Diagnosis for problem* $\mathcal{DP}(A_i)$ *is a Candidate Diagnosis:*

$$\Delta = \delta_1, \ldots, \delta_m$$

s.t.:

1. *all executions of IP process* $\mathcal{IP}$ *that contain sequence* Δ *eventually violate* cid
2. *for each prefix of* Δ*, there exists at least one execution that does not violate* cid

6 Computing Diagnoses

Before presenting the diagnostic algorithm, we must discuss how the Local Policy processes LP invoked by the agent processes $\mathcal{A}$ are modeled. Local Policy processes are problematic because, on the one side, the agent A_i performing a diagnosis needs a global point of view; on the other side, each agent desires to keep its local policies private. To cope with this problem we propose to implement both a private and a public view of Local Policies.

Definition 7 *A Local Policy Skeleton* $LP(pch)$ *is a* π*-calculus process that is either a* private *or a* public *representation of a LP of a role:*

- **priv***: is any implementation that computes the output* y *as a function of the input* $(x_1, \ldots, x_k)$[3]*:*

$$LP(pch) \stackrel{def}{=} pch(x_1, \ldots, x_k).\\ (determine\ y\ from\ x_1, \ldots, x_k).\overline{pch}(x_1, \ldots, x_k, y)$$

- **pub***: is an implementation that chooses its output nondeterministically from a predefined set* $\{y_1, \ldots, y_l\}$*:*

$$LP(pch) \stackrel{def}{=} pch(x_1, \ldots, x_k).\\ (\overline{pch}(x_1, \ldots, x_k, y_1) + \ldots + \overline{pch}(x_1, \ldots, x_k, y_l))$$

Thus, to guarantee the autonomy and privacy of agents that participate in the protocol execution, each agent A_i has its own specific representation $\mathcal{IP}_i$ of the Interaction Protocol. In particular, the internal policies of A_i are modeled in $\mathcal{IP}_i$ through fully specified *priv* LP processes, while the internal policies of the other agents, that are unknown to A_i, are modeled through *pub* processes. A Diagnostic Problem submitted to A_i will therefore be solved using $\mathcal{IP}_i$ as the IP model.

The algorithm adopted by A_i to solve the problem is depicted in Figure 3. It takes as inputs the diagnosing agent A_i, its π-calculus model $\mathcal{IP}_i$ of the IP, the identifier cid of the commitment $cc(A_i, A_j, p, q)$ violated by A_i, and the sequence σ_{DP} which is the ordered merge of the traces σ_i and $\sigma_{i,j}$s (see section 5).

[3]In general, more than one value y may be derived given an input $(x_1, \ldots, x_k)$, depending on other state variables. This point does not impact the current discussion.

Diagnose(A_i, $\mathcal{IP}_i$, cid, σ_{DP})
1. $\Delta \leftarrow ()$
2. **while not** done? **do**
3. $\quad\delta \leftarrow$ head(σ_{DP})
4. $\quad\sigma_{DP} \leftarrow$ tail(σ_{DP})
5. $\quad$pol? $\leftarrow$ ($\delta = lp(x_1, \ldots, x_k, y)$)
6. $\quad$viol? $\leftarrow$ ($\delta = cid'(\texttt{viol})$ **and** cred(cid') = A_i)
7. $\quad$rcv? $\leftarrow$ ($\delta = ch(v_1, \ldots, v_k)$)
8. $\quad$handle? $\leftarrow$ pol? **or** viol? **or** rcv?
9. $\quad$**if** handle? **then**
10. $\quad\quad\Delta \leftarrow \Delta \cdot \delta$
11. $\quad\quad$done? $\leftarrow$ Check($\mathcal{IP}_i$, Δ, cid)
12. $\quad$**end if**
13. **end while**
14. **return** Δ

Figure 3: The diagnostic algorithm.

First of all, the diagnosis Δ is set to the empty list. The flow then enters a *while* loop that will consume one action δ at a time from σ_{DP}, until the diagnosis has been found (i.e., flag $done?$ has been set to $\top$).

After updating σ_{DP} (by removing δ), the algorithm computes the values of three boolean expressions, that correspond to the three cases of Definition 5 in which δ should be added to the diagnosis Δ. More specifically: $pol?$ is $\top$ iff δ is a receive from an internal policy process; $viol?$ is $\top$ iff δ is the violation of a commitment that some agent has made to A_i; and $rcv?$ is $\top$ iff δ is a receive from some other agent. If any one of the flags is $\top$, we must handle the action δ ($handle?$ flag set to $\top$).

Just because δ has one of the forms listed in Definition 5, it must surely be part of the diagnosis, so δ is added to Δ. It is less obvious whether the updated Δ satisfies condition (1) of Definition 6, i.e., whether all executions of $\mathcal{IP}_i$ with a prefix containing sequence Δ lead to a violation of cid. In order to answer this question, the algorithm calls a procedure *Check*, which returns $\top$ iff condition (1) is satisfied. It is easy to see that, in such a case, the current value of Δ represents the diagnosis for the given problem, thus the $done?$ flag is set to $\top$, and the algorithm exits the *while* loop returning Δ.

The *Check* procedure is implemented through a call to an external Model-Checker. In our current implementation, we have used the SPIN Model-Checker [Holzmann, 1997]. Since SPIN requires that the system model is expressed in the PROMELA language, first of all we translate our π-calculus processes in PROMELA. Given the expressive power of PROMELA, this can be done quite straightforwardly, as explained in [Song and Compton, 2003].

The SPIN Model-Checker is able to verify the validity of LTL (Linear Temporal Logic) formulas w.r.t. PROMELA system models. In particular, the truth of an LTL formula φ is evaluated w.r.t. a state s of a (possibly infinite) sequence of states $\Sigma = (s_0 s_1 \ldots)$, where each state is an evaluation of a set of atomic propositions AP. The formula φ consists of the atomic propositions in AP combined with the usual propositional connectives $\wedge$, $\vee$, $\neg$, plus some *modal* operators (see Table 1).

Given the PROMELA translation $\mathcal{IP}^P$ of a π-calculus model $\mathcal{IP}$, we can ask SPIN to check whether an LTL formula φ is valid, i.e. if it is true in the initial state s_0 of *all* sequences Σ corresponding to the possible runs of the

$\bigcirc\varphi$	(*next* φ) holds in s iff φ holds in the state s' next to s in Σ
$\Box\varphi$	(*always* φ) holds in s iff for each s' that comes after s in Σ: φ holds in s'
$\Diamond\varphi$	(*sometime* φ) holds in s iff for some state s' that comes after s in Σ: φ holds in s'

Table 1: LTL Modal Operators

system $\mathcal{IP}^P$. The formula φ that we need to check can be semi-formally expressed as follows:

$$\Box((\Delta \; has \; happened) \rightarrow \Diamond(commitment \; cid \; is \; violated)) \quad (1)$$

where Δ and cid are, respectively, the (partial) diagnosis and the commitment passed to the *Check* procedure. The above sentence can be read as: if a state s is reached after the sequence of actions Δ have happened, then the run eventually reaches a state s' where cid is violated. It is easy to see that, if this is guaranteed to happen for any run of $\mathcal{IP}^P$, Δ is the diagnosis.

In order to encode Δ in an LTL formula, we must consider that during the translation of the $\mathcal{IP}$ model to PROMELA, we introduce the following state variables:

- for any action $lp(in_1, \ldots, in_k, out)$: variables lp_in_1, ..., lp_in_k, lp_out, set with the values received from channel lp
- for any action $ch(in_1, \ldots, in_k)$: variables ch_in_1, ..., ch_in_k, set with the values received from channel ch
- for any action $cid'(\texttt{viol})$: a variable cid' set to name `viol`

In other words, we introduce a set of state variables for describing the effects of the actions that appear in Δ. We can then define the antecedent $\Phi(\Delta)$ of the semi-formal LTL formula (1) recursively as follows:

$$\Phi(\Delta) = \begin{cases} lp\_in_1 = u_1 \wedge \ldots \wedge lp\_out = v & \Delta = lp(\vec{u}, v) \\ ch\_in_1 = u_1 \wedge \ldots \wedge ch\_in_k = u_k & \Delta = ch(\vec{u}) \\ cid' = \texttt{viol} & \Delta = cid'(\texttt{viol}) \\ \Phi(\delta) \wedge \Diamond(\Phi(\Delta')) & \Delta = \delta \cdot \Delta' \end{cases}$$

The definition states that, if Δ only contains an action, the antecedent $\Phi(\Delta)$ is a set of equalities that encode the action using the state variables introduced above. If, on the other hand, Δ is an action δ followed by a suffix Δ', then $\Phi(\Delta)$ consists of the encoding of δ conjoined with the request (expressed with $\Diamond$) that eventually also the rest Δ' will happen.

The LTL formula (1) above is then refined to the following:

$$\Box(\Phi(\Delta) \rightarrow \Diamond(cid = \texttt{viol}))$$

By evaluating this formula with SPIN, we perform the *Check* on Δ required by the diagnostic algorithm.

Example 2 *Let us go back to the bookstore scenario, and consider the commitment*

$$c_{DEL} = cc(BS, CS, pay(\texttt{hp7}), deliver(\texttt{hp7}))$$

existing between bookstore BS and customer CS. Let us now assume that the commitment state evolves to `violated`*. Of course, CS, being the creditor, wants to know why the commitment has been violated. The bookstore, being the debtor of the violated commitment, is in charge of solving the following diagnostic problem:*

$$\mathcal{DP}(BS) = (\mathcal{IP}_{BS}, c_{DEL}, \sigma_{BS}, \{\sigma_{BS,CS}\})$$

Of course, the diagnosis depends on the content of the two traces $\sigma(BS)$ and $\sigma_{BS,CS}$. In this first case, we assume that the bookstore is directly in charge of the delivery; it follows that BS interacts only with CS, and only the trace $\sigma_{BS,CS}$ in the KB shared between these two agents has to be considered. The two traces are merged into a single trace σ_{DP} where event ordering is maintained, and let us suppose that such a trace contains:

$$\begin{array}{l} pay(\texttt{hp7}).\overline{po\_pol}(\texttt{hp7}).po\_pol(\texttt{hp7}, \texttt{wait}). \\ \overline{po\_pol}(\texttt{hp7}).po\_pol(\texttt{hp7}, \texttt{wait}).\widehat{deliver}(\texttt{no}) \end{array}$$

Following the algorithm in Figure 3, the search for a diagnosis Δ consists in finding the minimal prefix of σ_{DP} that entails the violation of the commitment. Thus the first Candidate Diagnosis that is Checked *is Δ=$\{pay(\texttt{hp7})\}$, which is translated into the following LTL formula :*

$$\Box((pay\_in_1 = \texttt{hp7}) \rightarrow \Diamond(c_{DEL} = \texttt{viol}))$$

stating that, for all executions of $\mathcal{IP}_{BS}$, if a state satisfies $(pay\_in_1 = \texttt{hp7})$, then some later state will satisfy $cid_{DEL}(\texttt{viol})$. Since this initial candidate diagnosis does not entail the violation, the SPIN engine returns $\perp$, and the Candidate Diagnosis is incrementally extended by considering the next events in the σ_{DP} trace.

In this simple example, the only possible explanation is to consider the trace σ_{DP} up to the second `wait`*, and conclude that the cause for the commitment violation is due to the BS internal policy. More specifically, the candidate that turns out to be the actual diagnosis is:*

$$pay(\texttt{hp7}).po\_pol(\texttt{hp7}, \texttt{wait}).po\_pol(\texttt{hp7}, \texttt{wait})$$

which is translated to the following LTL formula:

$$\begin{array}{l} \Box((pay\_in_1 = \texttt{hp7}) \wedge \\ \quad \Diamond((po\_pol\_in_1 = \texttt{hp7} \wedge po\_pol\_out = \texttt{wait}) \wedge \\ \quad\quad \Diamond(po\_pol\_in_1 = \texttt{hp7} \wedge po\_pol\_out = \texttt{wait})) \\ \quad \rightarrow \Diamond(c_{DEL} = \texttt{viol})) \end{array}$$

Namely, the BS's choice of waiting twice for more books to be delivered has caused the violation of the commitment c_{DEL}.

Let us now consider a different scenario where BS relies on a shipper SH for actually delivering the ordered books. In this case two commitments are created. The first is c_{DEL} as before, the second has the following shape:

$$c'_{DEL} = cc(SH, BS, orderdel(\mathbf{g}), deliver(\mathbf{g}))$$

In this scenario, as soon as BS gets payed for a book **g***, it orders the delivery of the book to the shipper SH, that will be in charge of the delivery to the customer. The bookstore is still committed towards the customer that, whenever a payment is made, the requested object is delivered (commitment c_{DEL}), but its internal policy no longer requires it to wait for more objects to be delivered. We assume, however, that SH has some policy that makes that particular delivery fail.*

In such a case, the customer CS will complain with BS that `hp7` *has not been delivered as expected: c_{DEL} violated as before. Again, BS is in charge of solving a diagnostic problem since it is the debtor of an observed violated commitment. The diagnostic problem now has the following shape:*

$$\mathcal{DP}(BS) = (\mathcal{IP}_{BS}, c_{DEL}, \sigma_{BS}, \{\sigma_{BS,CS}, \sigma_{BS,SH}\})$$

In this case, BS also considers the KB shared with SH, and combining the three traces included in the problem it yields the following trace:

$$pay(\texttt{hp7}).\widehat{orderdel}(\texttt{hp7}).c'_{DEL}(\texttt{act}).c'_{DEL}(\texttt{viol})$$

In this case the only possible diagnosis is that the violation of c_{DEL} is indeed an indirect consequence of the violation of c'_{DEL}, i.e. the actual diagnosis is:

$$pay(\texttt{hp7}).c'_{DEL}(\texttt{viol})$$

From the point of view of BS, however, it is not apparent why SH has not delivered the book; this is because the internal policy of SH is hidden to all the other agents. The bookstore can however now complain with the shipper, and can inform the customer that the shipper should be blamed as responsible for the violation of c_{DEL}.

7 Conclusions and Future Work

Many alternative approaches have been proposed in literature to cope with the diagnosis of Multi-Agent Systems. A first family of approaches is based on *agent fault models* (e.g., [Dellarocas and Klein, 2000; Micalizio and Torasso, 2014]) which focus on agents' failures and their effects throughout the system, but do not consider the case of erroneous interactions. Coordination failures have been taken into account by *social diagnosis* [Kalech, 2012], which assumes that agents are cooperative and willing to share their beliefs; in open MAS this assumption does not hold in general. An alternative approach is the extension of Spectrum-based Fault Localization for MAS (ESFL-MAS) proposed in [Passos *et al.*, 2015]. The ESFL-MAS approach is model-less and localizes faulty agents by evaluating a *similarity coefficient* between the current run of the system and the history of past, failed runs of the same system. The advantage of this approach is that it does not require to define an explicit model of the system, which is instead implicitly represented by the historical data, not always available, though, in real-world applications.

In this paper, we have presented a formalization of the diagnosis of commitment violations in the execution of Interaction Protocols. While there exist several works on the diagnosis of classic workflow-based systems (e.g., [Verbeek *et al.*, 2001]), for commitment-based IPs most previous proposals only deal with the system monitoring and detection of violations [Robinson and Purao, 2009; Kafalı and Yolum, 2009]. To the best of our knowledge, the main exception is represented by [Kafalı and Torroni, 2012], where the authors propose a method for diagnosing misalignments in the representation of the commitments between the debtor and the creditor. Beside misalignments, their approach is also able to diagnose *misbehaviors* that can be either violations that have no further explanations, or wrong delegations of commitments from one debtor-creditor couple (x_1, y_1) to a couple of delegates (x_2, y_2). Thanks to a π-calculus model that associates agents behavior with the outcomes of decisions made by applying internal policies, our proposal is able to yield more informative explanations, without disclosing to the agent in charge of diagnosis the internal workings of the policies of other agents.

We regard this proposal as a first step that opens the way to the development of several further techniques for the "rich" diagnosis of violations in commitment-based Interaction Protocols. First of all, as noted in the paper, we currently deal only with atemporal commitments, while realistic scenarios tend to associate deadlines (or even feasibility intervals) to both the antecedents and consequents of commitments; the fact that PROMELA is able to deal with numeric variables may be a useful feature towards this extension.

Another very important line of research would focus on making the diagnosis process more distributed: currently, if the agent A that is diagnosing a commitment cid observes the violation of another commitment $cid'(viol)$ in its trace, it may infer that such a violation made the violation of cid unavoidable, and return $cid'(viol)$ as the last element of the diagnosis of cid. One obvious extension would be to invoke the debtor of cid' to explain in detail why cid' was violated. It would be interesting to further extend this mechanism to query another agent A' also in case the violation of cid was made unavoidable by the reception of a message $ch(\vec{v})$ from A'. There is a whole other aspect of making the diagnosis process more distributed, which is more subtle but very interesting: when the diagnoser agent A invokes the Model-Checker, it uses its own model $\mathcal{IP}_i$ of the IP, which replaces the internal policies of other agents with their public implementations that allow any output given an input. This can lead to over-optimistic conclusions from A, which may think that it is still possible to satisfy the commitment cid even when that is not the case; this results in terminating the diagnostic process later, returning a diagnosis that is minimal given the knowledge of A, while it is not so from a global point of view.

A final line of further research may partially be an alternative to address this problem. One peculiarity of the proposed diagnostic algorithm compared to most MBD (Model-Based Diagnosis) algorithms is that it returns exactly one diagnosis. That is due to the fact that it does not try to make hypotheses for what it does not know: if, e.g., a message $ch(\vec{v})$ is received by another agent A', the agent A does not try to speculate about what *may* have happened within A' that caused A' to reply in that way; however, with public knowledge about the association between the agents behavior and their decisions, agent A may in fact try to infer something about the decisions made by A', even without asking A' to cooperate in the diagnosis. This would likely lead to the usual explosion of possible diagnoses typical of MBD; and it would likely need to be alleviated by the usual means, in particular preference criteria that rank hypotheses based on their likelihood.

Acknowledgments

This work was partially supported by the Accountable Trustworthy Organizations and Systems (AThOS) project [ATHOS, 2016], funded by Università degli Studi di Torino and Compagnia di San Paolo (CSP 2014).

References

[ATHOS, 2016] ATHOS. `http://www.di.unito.it/~micalizi/athos/`, 2016.

[Castelfranchi, 1995] Cristiano Castelfranchi. Commitments: From Individual Intentions to Groups and Organizations. In Victor R. Lesser and Les Gasser, editors, *Proceedings of the First International Conference on Multiagent Systems (ICMAS)*, pages 41–48, San Francisco, California, USA, June 1995. The MIT Press.

[Chopra and Singh, 2009] Amit K Chopra and Munindar P Singh. Multiagent commitment alignment. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 937–944. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[Dellarocas and Klein, 2000] Chrysanthos Dellarocas and Mark Klein. An experimental evaluation of domain-independent fault handling services in open multi-agent systems. In *4th International Conference on Multi-Agent Systems, ICMAS 2000, Boston, MA, USA, July 10-12, 2000*, pages 95–102, 2000.

[Desai *et al.*, 2005] Nirmit Desai, Ashok U. Mallya, Amit K. Chopra, and Munindar P. Singh. Interaction protocols as design abstractions for business processes. *Software Engineering, IEEE Transactions on*, 31(12):1015–1027, 2005.

[Fornara and Colombetti, 2003] Nicoletta Fornara and Marco Colombetti. Defining interaction protocols using a commitment-based agent communication language. In *Proceedings of the second international joint conference on Autonomous Agents and Multiagent Systems*, pages 520–527. ACM, 2003.

[Holzmann, 1997] Gerard J Holzmann. The model checker spin. *IEEE Transactions on software engineering*, 23(5):279, 1997.

[Kafalı and Torroni, 2012] Özgür Kafalı and Paolo Torroni. Exception diagnosis in multiagent contract executions. *Annals of Mathematics and Artificial Intelligence*, 64(1):73–107, 2012.

[Kafalı and Yolum, 2009] Özgür Kafalı and Pınar Yolum. Detecting exceptions in commitment protocols: Discovering hidden states. In *Languages, Methodologies, and Development Tools for Multi-Agent Systems*, pages 112–127. Springer, 2009.

[Kalech, 2012] Meir Kalech. Diagnosis of coordination failures: a matrix-based approach. *Autonomous Agents and Multi-Agent Systems*, 24(1):69–103, 2012.

[Micalizio and Torasso, 2014] Roberto Micalizio and Pietro Torasso. Cooperative monitoring to diagnose multiagent plans. *Journal of Artificial Intelligence Research (JAIR)*, 51:1–70, 2014.

[Milner, 1999] Robin Milner. *Communicating and mobile systems: the pi calculus*. Cambridge university press, 1999.

[Passos *et al.*, 2015] Lúcio S. Passos, Rui Abreu, and Rosaldo J. F. Rossetti. Spectrum-based fault localisation for multi-agent systems. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1134–1140, 2015.

[Reiter, 1987] Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.

[Robinson and Purao, 2009] William N Robinson and Sandeep Purao. Specifying and monitoring interactions and commitments in open business processes. *IEEE software*, 26(2):72, 2009.

[Singh, 1999] Munindar P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.

[Song and Compton, 2003] Hosung Song and Kevin J Compton. Verifying π-calculus processes by promela translation. *University of Michigan, Tech. Rep. CSE-TR-472-03*, 2003.

[Telang *et al.*, 2011] Pankaj R Telang, Munindar P Singh, and Neil Yorke-Smith. Relating goal and commitment semantics. In *Programming Multi-Agent Systems*, pages 22–37. Springer, 2011.

[Van Der Aalst and Van Hee, 2004] Wil Van Der Aalst and Kees Max Van Hee. *Workflow management: models, methods, and systems*. MIT press, 2004.

[Verbeek *et al.*, 2001] Henricus MW Verbeek, Twan Basten, and Wil MP van der Aalst. Diagnosing workflow processes using woflan. *The computer journal*, 44(4):246–279, 2001.