Under consideration for publication in Knowledge and Information Systems

PISAGOR: a ProactIve Software AGent for mOnitoring inteRactions

Özgür Kafalı[1] and Pınar Yolum[2]

[1]Department of Computer Science, North Carolina State University, Raleigh, NC, USA
[2]Department of Computer Engineering, Bogazici University, İstanbul, Turkey

Abstract. E-commerce has become an integral part of everyday life. Many e-commerce systems enable users to browse products, place orders, and track deliveries online. However, these interactions are time-consuming, especially if they do not proceed as expected. Instead of a human user, software can offer automated support to monitor e-commerce transactions to ensure that they are completed by the businesses as planned. Accordingly, this paper studies methods to develop a software agent for monitoring its users' interactions with businesses (PISAGOR). The interactions are represented as commitments, which have been extensively studied in multiagent systems. With a commitment-based specification at hand, the agent knows what the meaning of its interactions are and can therefore reason over its actions. The reasoning is accompanied with the as-good-as relation that compares the agent's current state with its expectations from a transaction. This enables PISAGOR to decide whether the transaction is progressing as expected. Moreover, we propose operational rules for the agent to create expectations based on its commitments and check its progress towards them. We demonstrate via a case study how PISAGOR can detect if a user's interactions are not progressing well and identify a problem for the user to take action.

Keywords: Multiagent systems; Commitments; Agent-based commerce; Expectations

1. Introduction

An important part of commerce is being carried online through e-commerce systems. These system have evolved considerably over the years to provide various functionalities to its users, such as easy user interfaces, auction mechanisms, negotiation, recommendation and reputation services, contract management, and so on (Özer and Özturan, 2011; Kersten, Vahidov and Gimon, 2013; Gupta and Chakraborti, 2013; Chang and Wong, 2011; Morid and Shajari, 2012; Wood, Kafalı and Stathis, 2013; Yan, Fong and

Received May 27, 2014
Revised Oct 28, 2014
Accepted May 07, 2015

Shi, 2008). As a result, current e-commerce systems work effectively and serve human users' needs well. However, many times there is still a need for a human to be the user of these systems. Ideally, it would be easier for humans if they could delegate their task of interacting with the system to specialized software.

A major paradigm for realizing such software is agents: software that can perceive and reason about the environment and act as a result of that (Singh and Huhns, 2005). Agents represent users and fulfill the tasks that are cumbersome for them. Among the tasks that are most time-consuming for the users is the task of monitoring the progress of transactions. Even though the software support is there to enable and verify transactions (Yao, Ruohomaa and Xu, 2012; Huang, Hua, Will and Wu, 2012), it is missing for monitoring transactions once they are initiated. However, there are many situations that need constant attention from users after a transaction is initiated.

Consider the following workflow for selling a house on the market. The landlord goes to an estate agent, gets a quote for the house's value and is advised to refurbish the house before putting it on the market. The landlord then goes to a contractor to arrange the refurbishment. The contractor first needs to get the payment from the landlord to acquire the necessary materials from a supplier and the required tools from a manufacturer. The contractor will then hire a builder to do the job. Once the refurbishment is done, the estate agent can then sell the house for a good price. If everything goes as planned, the landlord expects to sell the house within two months.

Our aim in this work is to develop an agent that can reason about the user's circumstances and detect unexpected situations. To do this, the agent would need to know the contracts between the user and others, and create expectations based on the fulfillment of the tasks that are specified by the contracts. If the agent is aware of the contracts and expectations, it should be able to decide if the evolution of the interactions is aligned with the expectations of its user. If there is discrepancy between the expectations and the real life situation, then the agent would warn the user to take an action to deal with this.

Example 1 demonstrates a detailed scenario for the above case and outlines possible monitoring operations that can be performed.

Example 1. Bob wants to sell his house within two months. The estate agent tells him that if the house is refurbished first, then he can sell it for a good price in a month's time. So, Bob meets with a contractor to see if the refurbishment can be done in a month. The contractor requires the payment from Bob in advance to acquire the materials and tools required for the job. Bob sends the payment in a week to make sure the job is completed by the end of the month.

Following this example, the ideal situation is that the contractor delivers the task by the end of the month. However, this might not always be the case. Various other things might happen instead:

- Bob might not be in touch with the contractor during the refurbishment process and it might be delayed due to a problem with the builder that the contractor has hired. This would be a violation of the contract on the contractor's side and would clearly not meet Bob's expectations. This is a bad outcome since the fact that the expectation is not being met is only realized on the day of the deadline.
- The contractor could check the delivery status of the materials on a daily basis. If they are not dispatched soon, he can decide to cancel the order and reorder from a different supplier.
- Assume that Bob hears of news that the house prices in his area will go up in six months due to new offices being built. Although the refurbishment is delayed and the

house cannot be sold within two months, this would not be an exception for Bob as he would want to wait until the house prices go up.

Ideally, the monitoring that is involved in the above cases could all be done with a software agent. Moreover, an agent can tell the user whether there is a situation that needs to be taken care of or whether the agent (or the user) needs to take action. For example, the agent for the contractor itself can send a reminder to the landlord if the payment is late.

This paper develops a principled, distributed approach to enable agents to specify the agreements that their users are in, represent their expectations, and more importantly compute whether their expectations are attainable via the current progress of the agent in the system. Our proposed approach respects agents' privacy and subjectivity in that each agent is only aware of its own user's status and expectations. The specific contributions of this paper are as follows:

- We incorporate the well-studied commitment theory from multiagent systems (Torroni, Yolum, Singh, Alberti, Chesani, Gavanelli, Lamma and Mello, 2009; Kafalı and Yolum, 2011; Chesani, Mello, Montali and Torroni, 2013) to represent an agent's interactions and agreements with others. These commitments capture the participants' transactions.
- In order to relate agents' current states with their expectations, we propose the *as-good-as* relation. In addition to the satisfiability relation in propositional logic, this relation also compares stateful commitments with deadlines. If the current state of the agent is not as-good-as its expected state, we conclude that there is an action that needs to be taken.
- We propose operational rules that the agent uses to create and update expectations based on the state of its commitments and check its progress towards them.
- We develop the PISAGOR agent for monitoring interactions. To enable this, we have extend the Reactive Event Calculus ($\mathcal{REC}$) (Chesani et al., 2013) to implement our as-good-as relation.
- We analyze a case study to show that PISAGOR can successfully signal the right places for its user to take an action, similar to the ones listed above.

The rest of the paper is structured as follows. Section 2 describes our general multiagent architecture. Section 3 explains our formalization. Section 4 formalizes the *as-good-as* relation, and Section 5 describes how it is used by PISAGOR to operate. Section 6 explains our implementation and demonstrates how our approach works on a case study. Finally, Section 7 concludes the paper by reviewing relevant literature, summarizing our contributions, and proposing possible future directions.

2. Architecture

We focus on e-commerce systems where the sellers and buyers are bound with an agreement (similar to a contract) (Jakob, Pěchouček, Miles and Luck, 2008; Telang and Singh, 2012*a*; Wood et al., 2013), such that each clause in the agreement would describe a certain transaction between the participants. Each party in the system (buyer, seller, and so on) is represented with an agent that manages its user's interaction.

Figure 1 shows an initialization of our architecture with our previous scenario in mind. Note that each agent has access to a portion of the protocol that is relevant for itself. This is to ensure autonomy and privacy in the system. Similarly, expected states

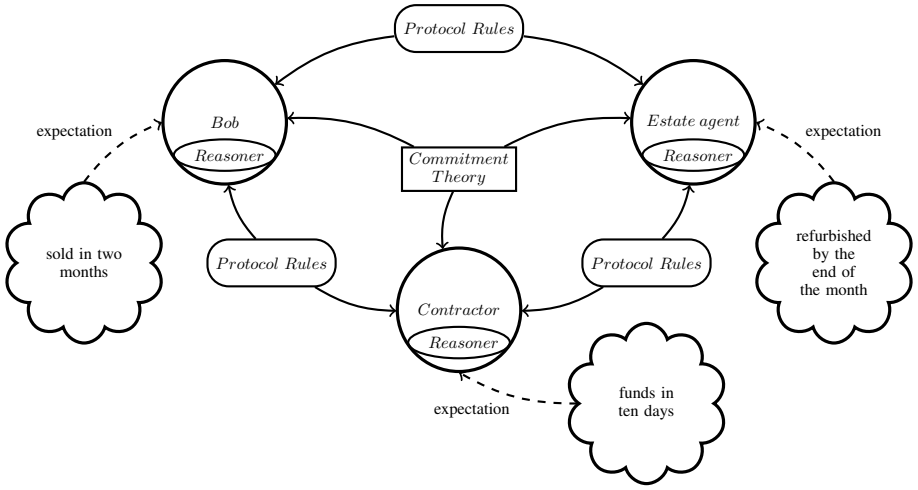


Fig. 1. Pictorial representation of the scenario (Example 1) and its components that follows a distributed multiagent architecture. Circle nodes denote the agents that represent Bob, the estate agent and the contractor. Each agent enacts a protocol, which is shown with a rounded rectangle. At the core of the system is the commitment theory shown with a rectangle, which formalizes the agents' interactions. Each agent has an expected state depicted by a cloud, and a separate reasoner to monitor its interactions. The arrows show which information is fed into the agent's reasoner.

are private to agents, and they denote what the agent expects from its transactions as an outcome. For example, the expectation on the left is only connected to the related agent Bob, whereas the protocol description on the top is connected to the two related agents Bob and the seller. Each agent has a reasoner that enables it to infer whether expected states are reached or not.

Overall, the agents enact a distributed protocol described by a set of interaction and commitment manipulation rules, and they constantly check if the transactions are progressing as planned. To do so, an agent compares its expectations with its current state of the world at a given point in time. Agents may have different expectations about their future. By sensing the differences between their expectations and the reality they face, they should be able to signal problems, recover from them, and continue proper working.

Below, we describe each element in more depth and then give a sample execution to demonstrate the usage of the overall system.

Protocol rules: A protocol is described by a set of rules defining the consequences of agents' actions (Mallya and Singh, 2007; Kafalı and Yolum, 2011). Following a distributed execution, protocol descriptions are shared among related agents, so that an agent only has access to protocol rules concerning itself. This is important for preserving *privacy*. In real life, each agent would be aware of the protocol that it is involved in, but would not know the agreements among other parties. In addition, by coupling the protocol descriptions among agents, we ensure that they share the same semantics for the protocol rules (e.g., infer the same consequences for actions), thus providing *interoperability*.

Commitment theory: Commitments are an important abstraction for understanding

and reasoning about agent interactions (Torroni et al., 2009). A commitment captures a promise from a debtor to a creditor to bring about a condition. They enable flexible interactions among agents. The key idea in modeling agent interactions with commitments is that messages are given a meaning based on creation and manipulation of commitments. That is, each message among agents either commits an agent to bring about a proposition or alters one of the existing commitments of an agent. Knowing the meaning of each message in the system empowers the agents to decide on their actions on their own. The commitment theory in the architecture defines the rules for creating and manipulating commitments. In our architecture, the commitment theory is shared among all the agents. This is required so that all agents can manipulate their commitments in the same way and to prevent commitment misalignments from occurring (Chopra and Singh, 2009; Kafalı, Chesani and Torroni, 2010). We describe the formal theory abut commitments in Section 3.

Expectations: Each agent may have several expectations about its future. For example, after transferring the funds, Bob expects the house to be refurbished by the end of the month. We develop the theory on expectations in Section 3 using agent states.

Events: Events correspond to the actions of the agents as well as any external activities. For example, *transfer_funds* is an event. Each event is associated with a time point that describes at which time the event has occurred. Certain properties of the system are initiated or terminated as consequences of events. We exemplify the usage of events in the case study (see Section 6).

Reasoner: Each agent has a separate reasoner that it uses to check whether its expectations are being met. The reasoner is fed with the protocol rules, the commitment theory, the agent's expectations, and a sequence of events that are significant to the agent's execution. If its expectations are not being met, the agent creates a warning for the user to take an action. We describe how reasoning in PISAGOR is performed in Section 5 via the as-good-as relation.

Now, let us see how Example 1 can be connected with the above elements to identify a problem for Bob regarding his intention to sell the house in two months. The protocol is described by the following actions:

- Offer (contractor): The contractor offers Bob to refurbish his house.
- Transfer funds (landlord): Bob sends the payment to the contractor.
- Send materials (supplier): The supplier sends the materials to the contractor.
- Send tools (manufacturer): The manufacturer sends the tools to the contractor.
- Refurbish (builder): The builder completes the refurbishment of the house.
- Sell house (estate agent): The estate agent sells the house.

In this work we provide a systematic way for the users to decide when there is something wrong regarding their interactions. Note that the process afterwards is not within the scope of this work, e.g., Bob might switch to a new contractor if refurbishment is not performed on time. The following trace demonstrates a possible execution of the protocol:

1. The contractor makes the offer (event),
2. A contract has been created telling that the contractor will refurbish the house within a month if Bob transfers the funds in ten days (protocol description),
3. Bob transfers the funds (event),
4. Following timely payment, the contractor becomes committed to Bob (commitment theory),

5. Based on the contract, Bob expects the house to be refurbished by the end of the month (expectation),
6. Bob checks his progress to see whether his expectation is satisfied, and finds out it is not since refurbishment is delayed. Based on this finding, he identifies a problem (reasoner).

In order to detect such cases, we need a reasoner that can interpret our commitment theory and compare agent states to decide if the current state is as-good-as the expected state. In order to realize this reasoning, we develop a formal model of the as-good-as relation in terms of propositions, commitments and states.

3. Formal Model

Our formal model is based on the description of the world through states and the evolution of the states through agents' commitments (Kafalı and Yolum, 2011; Torroni et al., 2009; Chesani et al., 2013).

Each agent views a part of the world since the execution of the multiagent system is distributed among the agents. Hence, a state is subjective to an agent and captures the agent's view-point at a given point in time.

Definition 1. A time point is a discrete measure of time and is totally ordered. We use $t_1, t_2, \ldots, t_n$ to denote time points.

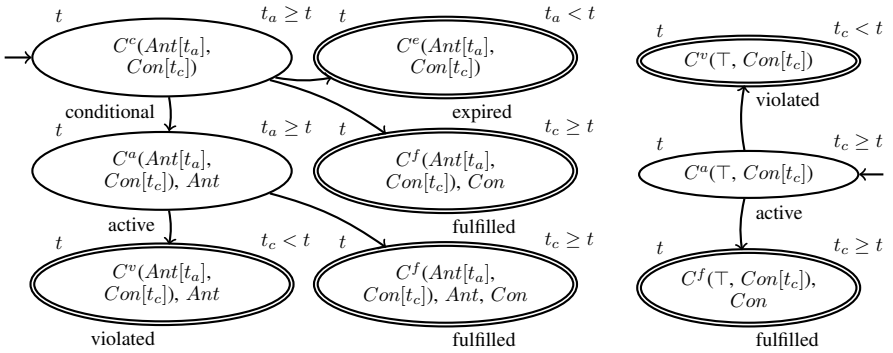
A state consists of propositions that are known to be true at that time point and commitments that are represented via their states. Propositions tell what has happened in the system so far (i.e., facts) to the agent's perception. We assume that the system is monotonic. Thus, once a proposition becomes true, it cannot become false.

Definition 2. A proposition can be an atomic proposition, its negation or the conjunction of atomic propositions.

A commitment represents a contract from a debtor agent towards a creditor agent about a specific property (Torroni et al., 2009). Definition 3 defines a commitment formally. Below, A_d and A_c denote agents; Ant and Con are propositions.

Definition 3. $C^{St}_{A_d,A_c}(Ant[t_a], Con[t_c])$ denotes the commitment between the agents A_d and A_c, such that if the antecedent Ant is satisfied (i.e., becomes true until the deadline t_a), then the debtor A_d becomes committed to the creditor A_c for satisfying the consequent Con until the deadline t_c (Torroni et al., 2009; Yolum and Singh, 2007; Mallya and Singh, 2007). St denotes the state of the commitment.

Consider the following commitment from the contractor (cnt) to Bob about refurbishing the house: $C^{St}_{cnt,bob}(funds[10], refurbished[30])$, where $funds$ means that the funds is transferred to the contractor's bank account and $refurbished$ means that the house is refurbished. The commitment states that if Bob transfers the funds in ten days, then the contractor will make sure the house is refurbished by the end of the month. When the commitment is first created, it is in conditional state, denoted $C^{c}_{cnt,bob}(funds[10], refurbished[30])$. At this point, neither $funds$ nor $refurbished$ hold. If $funds$ starts to hold before the deadline, then the contractor will be committed to bring about $refurbished$. We call this state active and denote it as $C^{a}_{cnt,bob}(funds[10], refurbished[30])$. If $funds$ does not hold before the deadline, then the commitment is expired $C^{e}_{cnt,bob}(funds[10], refurbished[30])$. If the commitment becomes active



(a) Conditional (b) Base-level

Fig. 2. Lifecycle of a commitment extended with deadlines. A commitment can be created as conditional or base-level. Double ellipses represent terminal commitment states (i.e., the commitment's life-cycle ends in those states).

and $refurbished$ is not realized before the deadline, then the commitment is violated $C^v_{cnt,bob}(funds[10], refurbished[30])$. This is the case where Bob has transferred the funds, but the house is not refurbished on time. When the commitment is conditional or active, if $refurbieshed$ starts to hold before the deadline, then the commitment is fulfilled $C^f_{cnt,bob}(funds[10], refurbished[30])$.

In certain cases, a commitment can be created without a precondition. That is, Ant is set to be $true$ ($\top$). In such cases, we say that the commitment is already active since there is nothing to be realized to make the precondition true.

Figure 2 summarizes the commitment lifecycle. Whenever the debtor and creditor are not relevant to the discussion, we remove them from the commitment notation to reduce clutter. We formally describe each commitment state in Section 3.1.

Next, we describe agent states.

Definition 4. A term $\mathcal{T}$ is either a proposition ϕ or a commitment C.

Definition 5. A state contains terms that hold at a particular time point T (Kafalı and Yolum, 2011; Mallya and Singh, 2007). To keep processing simple, we partition propositions Φ and commitments $\mathcal{C}$ in two different sets. Hence, $\mathcal{S}^T(X) = \langle \Phi, \mathcal{C} \rangle$.

We differentiate between two types of states:

- Current state: $\mathcal{S}^T(A)$: Each agent A has a local state (i.e., its local world model), which is a subset of the real world at time T. In a distributed execution, agents perceive the real world from different view-points. Thus, their states may differ from each other based on their perceptions.
- Expected state: $\mathcal{S}^T(A_p)$: In order to enable agents to reason about the future, it is important to represent how the agent expects the future. That is, based on its current state and its current commitments in this state, how does the agent expect the world to evolve at a future time point? We describe this with the notion of an expected state, which represents an expectation of A for the world at time T. The expected state is a representation of what the agent expects when time has evolved to that point.

Figure 3 shows examples for different types of states for the same time point 25.

(a) $\mathcal{S}^{25}(bob) = \langle\{funds\}, \{\mathcal{C}^{a}_{cnt,bob}\ (funds[10], refurbished[30])\}\rangle$

(b) $\mathcal{S}^{25}(estate\_agent) = \langle\{\}, \{\}\rangle$

(c) $\mathcal{S}^{25}(bob_p) = \langle\{refurbished\}, \{\}\rangle$

Fig. 3. Sample states.

In (a), Bob's actual state is shown. It contains a single proposition, which tells that the funds has been transferred, and an active commitment from the contractor to Bob, which tells that the refurbishment is still in process. Bob fully perceives its environment at this point. However, it is not always the case that the agents can perceive everything in their environment. For example, in (b), the estate agent is neither aware of the commitment between Bob and the contractor, nor aware of the fact that Bob has transferred the funds. In (c), Bob's expected state is shown. One can easily see that Bob expects refurbishment to be completed for this time point. However, from the local state of Bob, we see that the commitment for refurbishment is still active. Indeed, there is an discrepancy between what Bob perceives and expects.

3.1. Commitment Theory

We now describe each commitment state with respect to the corresponding world states. Assume the following state $\mathcal{S}^t(A) = \langle\Phi, \mathcal{C}\rangle$ and $C^{St}(Ant[t_a], Con[t_c]) \in \mathcal{C}$. Let's walk through different values of St (see also Figure 2).

Conditional: When the commitment is in conditional state, denoted $C^c(Ant[t_a], Con[t_c])$, if Ant is brought about by the deadline t_a, then the debtor will be committed to bring about Con by the deadline t_c. When the state is conditional, then Ant, $Con \notin \Phi$. In other words, the commitment $C^c(Ant[t_a], Con[t_c])$ cannot coexist in the same state with its antecedent or consequent:

- If the commitment's antecedent Ant already holds before t_a, then the commitment is no longer *conditional* (i.e., its condition is satisfied), and it will become *active*.
- If the commitment's consequent Con already holds before t_c, then the commitment is no longer *conditional*, and it will become *fulfilled*.

Expired: The commitment is in expired state, denoted $C^e(Ant[t_a], Con[t_c])$, when $C^{St}(Ant[t_a], Con[t_c]) \in \mathcal{C}$, $Ant \notin \Phi$ and $t_a < t$. That is, the antecedent of the commitment has not been brought about before its deadline has passed. The conditional commitment's life-cycle ends with this state. Note that we do not use the expired commitment state in this paper.

Active: A commitment can be in an active state following two different paths:

- The commitment is initially created without a condition, denoted as $C^a(\top, Con[t_c])$. We require that $Con \notin \Phi$. In other words, the commitment $C^a(\top, Con[t_c])$ cannot coexist in the same state with its consequent. If the commitment's consequent Con already holds before t_c, then the commitment is no longer *active*, and it will become *fulfilled*.

– Alternatively, a conditional commitment may be detached into an active commitment, when $C^{St}(Ant[t_a], Con[t_c]) \in \mathcal{C}$, $Ant \in \Phi$ and $t_a \geq t$. That is, the antecedent of the commitment has been brought about before its deadline.

Fulfilled: The commitment is in fulfilled state, denoted $C^f(Ant[t_a], Con[t_c])$, when $C^{St}(Ant[t_a], Con[t_c]) \in \mathcal{C}$, $Con \in \Phi$ and $t_c \geq t$. That is, the consequent of the commitment has been brought about before its deadline. The commitment's life-cycle ends with this state.

Violated: The commitment is in violated state, denoted $C^v(Ant[t_a], Con[t_c])$, when $C^{St}(Ant[t_a], Con[t_c]) \in \mathcal{C}$, $Ant \in \Phi$, $Con \notin \Phi$ and $t_c < t$. That is, the commitment has become (or started) active, but the consequent has not been brought about before its deadline has passed. The commitment's life-cycle ends with this state.

3.2. When to Take Action

A rational agent tries to perform actions that will enable its expected state to be realized. However, note that an expected state may not be realized solely by the agent itself since it may contain propositions or commitments that can only be satisfied by others. For example, even if Bob expects timely refurbishment of the house, he can't ensure it by himself. Given the current state, we are interested in computing if the current state yields a lower benefit for the agent than an expected state. This would mean that things are not progressing as planned and that an action needs to be taken. For example, Bob needs to take an action if the house is not refurbished on time.

Recall that a given state $\mathcal{S}^T(A)$ consists of terms that are either propositions or stateful commitments. We refer to all terms in a given state (that include Con) with $\mathcal{T}_{Con}(S)$ as described in Definition 6.

Definition 6. The set $\mathcal{T}_{Con}(S)$ contains all the terms in state S that are either a proposition Con, a proposition $\neg Con$, or a commitment whose consequent is the proposition Con.

The utility of a term $x \in \mathcal{T}_{Con}(S)$ shows how desirable x is in the execution. We assume a function u from terms into real numbers so that each agent can compute and compare the utilities of terms.

Definition 7. From the creditor's point of view, the following is meaningful: $u(\neg Con)$ = $u(C^v(Ant, Con)) < u(C^a(Ant, Con)) < u(C^f(Ant, Con))$ = $u(Con)$. For simplicity, deadlines are omitted here. The creditor would benefit most if the commitment is fulfilled. Less beneficial is an active commitment and the worst is a violated commitment. This ordering is compatible with the commitment valuations proposed by Yolum and Singh (Yolum and Singh, 2007).

Figure 4 demonstrates the condition where the agent will take an action. When an agent ends up achieving a term that has lower utility than its expected term, then this will yield a problem. For example, if the agent expects $C^f(Con)$, but in reality, the world has $C^a(Con)$, then it should take an action. Based on this intuition, we formally define when an agent needs to take an action in Definition 8. This definition captures the idea that the decision to take an action is agent-specific in the sense that one agent may identify a particular situation as a problem while another agent may not, depending on its expectations.

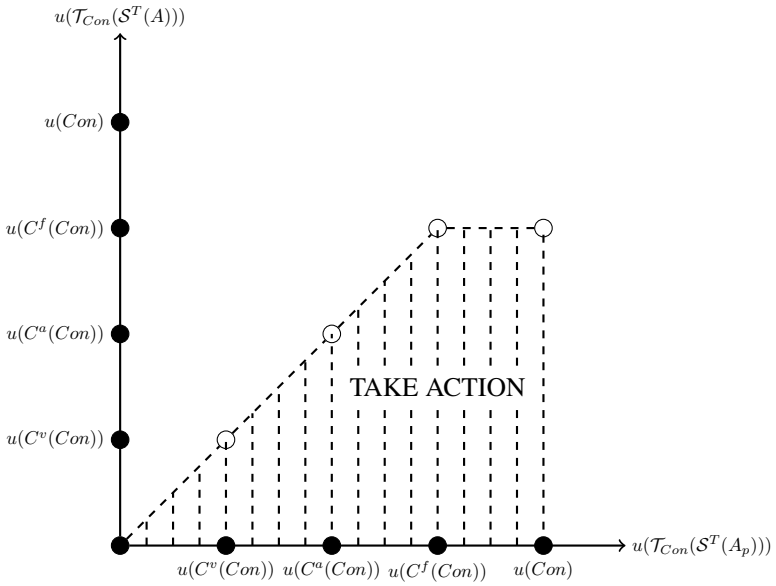


Fig. 4. Understanding when to take actions. The X axis shows expected terms, whereas the Y axis shows the actual terms. The dashed area shows the cases where the agent would take an action.

Definition 8. Agent A at time T needs to take an action related to proposition Con, denoted $takeAction(A, T, Con)$, iff

- the expected state contains a term but the real state does not contain any terms; that is: $\exists\, x$: $x \in \mathcal{T}_{Con}(\mathcal{S}^T(A_p))$ and $\mathcal{T}_{Con}(\mathcal{S}^T(A)) = \emptyset$, or
- the expected state contains a term that is higher in utility than the related term in the real state; that is: $\exists\, x, y$: $x \in \mathcal{T}_{Con}(\mathcal{S}^T(A_p))$, $y \in \mathcal{T}_{Con}(\mathcal{S}^T(A))$, and $u(x) > u(y)$.

Note that Definition 8 does not compare the total utility value of the states but of individual terms. The intuition is that we do not want the summation of utilities to cover for the differences in individual differences. For example, assume that there are two terms in the real state as well as in the expected state. It might be the case that when analyzed individually one of the terms in the real state has lower utility than its associated term in the expected state. Once there is a problem related to a single proposition in an agent's state, then the agent needs to take an action. Our definition follows this idea and signals a problem. However, if we had looked at the utilities of the states as a whole, it might have been the case that the utility gained from the second term is much higher in the real state, yielding a high utility for the real state than the expected state. Since the utility had been higher in the real state, the agent would not have identified a problem, missing a possible problem regarding the first term.

4. Comparing States with As-Good-As

We capture the expectations of an agent through expected states. On one hand, an expected state is similar to a temporal achievement goal (Chopra, Dalpiaz, Giorgini and

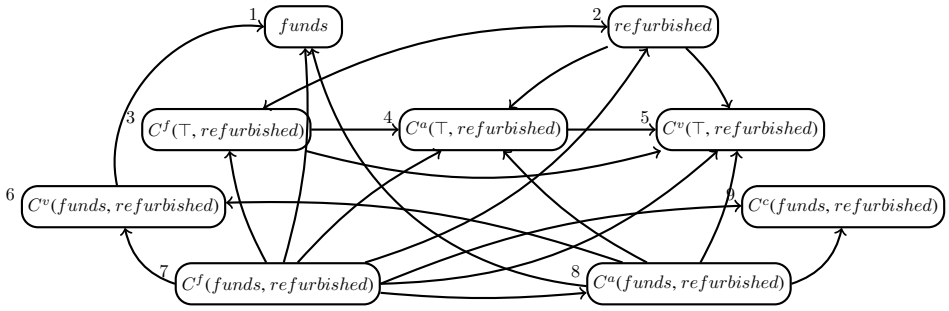


Fig. 5. A partial view of the "as-good-as network". Given two atomic propositions *funds* and *refurbished*, the network depicts some possible terms that can be derived from them and draws the as-good-as relation in between. Arrows show the direction of the relation. To reduce clutter, we remove the agents' names as well as the deadlines from the commitments.

Mylopoulos, 2010; Telang, Meneguzzi and Singh, 2013), where the agent plans to reach some properties or be involved in some commitments at a certain time point. On the other hand, expected states do not necessarily represent goals. They model the agent's expectations about the future, which might be good or bad. That is, the agent may assume certain properties to hold in the future even though it does not wish so, e.g., the contractor may expect a late delivery of the material if he is informed of a car accident covering the territory. Next we develop the theory for the *as-good-as* relation, that is used by PISAGOR to reason about its interactions.

It is important that an agent can compare its actual and expected states. For that purpose, we propose the *as-good-as* relation. It compares two states and decides if one is as-good-as the other. This is a strong relation in the sense that once a state is as-good-as another, then the two states are either equal or the former can replace the latter. This captures our intuition that if the current state of the world is equivalent to or better than the expected world state, the execution is in order and no action needs to be taken. However, if the comparison yields that the current state is not as-good-as the expected state, then there is a problem and it should be handled by the agent.

Our understanding of the as-good-as relation is inspired from satisfiability in propositional logic. However, in this case, we do not only have propositions but also commitments with states. Hence, we need a relation that can also take care of comparisons of stateful commitments.

Our as-good-as relation is denoted by $X \Vdash Y$, and is read as "X is as-good-as Y". Since the proposed relation will be used to compare the current state of an agent with an expected state, X will correspond to the real state and Y will correspond to the expected state of the agent.

We develop the as-good-as relation via the following axioms by extending and formalizing the ideas reported in (Kafalı and Yolum, 2011). Figure 5 is an aid in following the axioms. Note that the figure does not include deadlines for simplicity. In order to understand the relation between states, we start with studying the as-good-as relation between terms.

Axiom 1. Proposition ϕ_i is as-good-as proposition ϕ_j at time t, denoted $\phi_i \Vdash \phi_j$, iff $\phi_i \vdash \phi_j$.

Axiom 1 follows directly from logic entailment in propositional logic. When com-

paring propositions, we assume that they are described by the same time point, e.g., comparing the current state of the agent with its expected state for the same time point. When commitments are involved, we need to go beyond basic logic entailment. Accordingly, Axiom 2 states that if a proposition entails a commitment's consequent, then the proposition is as-good-as the commitment independently of its state.

Axiom 2. Proposition ϕ is as-good-as commitment $\mathcal{C}^S_{A_i,A_j}(Ant[t_a], Con[t_c])$ at time t, denoted $\phi \Vdash \mathcal{C}^S_{A_i,A_j}(Ant[t_a], Con[t_c])$, iff $\phi \vdash Con$ and $t_c \geq t$.

We can study Axiom 2 as four cases that correspond to the different commitment states $\{c, f, a, v\}$.

- A proposition entailing a conditional commitment's consequent is more beneficial than than the conditional commitment itself. The interpretation is that if the contractor expects that when he pays then the delivery takes place, and if the materials are actually delivered before him making the payment, then the current state of the world is more beneficial than his expectation. Hence, the proposition is as-good-as the fulfilled commitment.
- A proposition entailing a fulfilled commitment's consequent possesses the same outcome as the commitment itself. For example, in Figure 5, nodes 2 and 3 ($refurbished \Vdash C^f(\top, refurbished)$) exhibit this relation. The interpretation is that if Bob has an expectation that the commitment to him for refurbishment will be fulfilled and if he perceives that the house is indeed refurbished, then his expectation is satisfied. Hence, the proposition is as-good-as the fulfilled commitment.
- A proposition entailing an active commitment's consequent is more beneficial than the commitment itself. For example, in Figure 5, nodes 2 and 4 ($refurbished \Vdash C^a(\top, refurbished)$) exhibit this relation. That is, if Bob has an expectation that his commitment will be active (such that the refurbishment will take place later) but perceives that the house is already refurbished, then he is indeed in a better situation. Hence, the proposition is as-good-as the active commitment.
- A proposition entailing a violated commitment's consequent is more beneficial than the commitment itself. For example, in Figure 5, nodes 2 and 5 ($refurbished \Vdash C^v(\top, refurbished)$) exhibit this relation. Now, if Bob expects that his commitment will be violated (such that the house will not be refurbished on time), then he will be satisfied when he actually sees that the refurbishment is finished. Hence, the proposition is as-good-as the violated commitment.

Axiom 3. Commitment $\mathcal{C}^S_{A_i,A_j}(Ant[t_a], Con[t_c])$ is as-good-as proposition ϕ at time t, denoted $\mathcal{C}^S_{A_i,A_j}(Ant[t_a], Con[t_c]) \Vdash \phi$, iff

- $Ant \vdash \phi$ and $S \in \{a, v\}$, or
- $Con \vdash \phi$ and $S \in \{f\}$.

Axiom 3 describes the two ways that a conditional commitment is considered as-good-as a proposition. The first case states that if a conditional commitment reaches an active state, then it satisfies its antecedent. This is straightforward since the only way to reach an active conditional commitment is by having the antecedent hold. Further, if the consequent never becomes true (i.e., the commitment is violated), the commitment would still satisfy the antecedent; hence for example, in Figure 5, node 6 would satisfy node 1 ($C^v(funds, refurbished) \Vdash funds$). Note that the fulfilled state is not included in this case, because the conditional commitment may have been fulfilled directly from

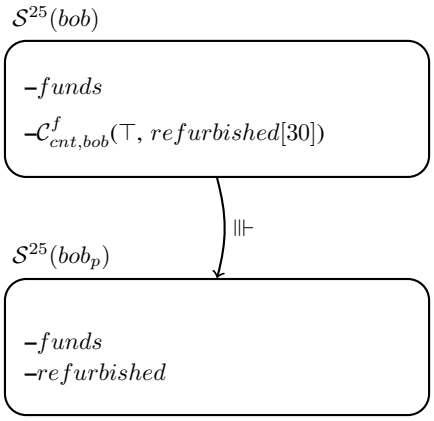


Fig. 6. As-good-as for states.

the conditional state leaving us with no information about the antecedent itself. The second case states that a fulfilled conditional commitment satisfies its consequent. In Figure 5, a similar example would be among nodes 7 and 2 ($C^f(funds, refurbished) \Vdash refurbished$).

Axiom 4. Commitment $\mathcal{C}^{S_1}_{A_i,A_j}(Ant_1[t_{a1}],\ Con_1[t_{c1}])$ is as-good-as commitment $\mathcal{C}^{S_2}_{A_k,A_l}(Ant_2[t_{a2}], Con_2[t_{c2}])$, denoted $\mathcal{C}^{S_1}_{A_i,A_j}(Ant_1[t_{a1}], Con_1[t_{c1}]) \Vdash \mathcal{C}^{S_2}_{A_k,A_l}(Ant_2[t_{a2}],$ $Con_2[t_{c2}])$, iff

- $Ant_2 \vdash Ant_1, Con_1 \vdash Con_2, S_1 \in \{f\}$ and $t_{c2} \geq t_{c1}$, or
- $Ant_2 \vdash Ant_1, Con_1 \vdash Con_2, S_1 \in \{a\}, S_2 \in \{a, v, c\}$ and $t_{c2} \geq t_{c1}$, or
- $Ant_2 \vdash Ant_1, Con_1 \vdash Con_2, S_1 \in \{v\}, S_2 \in \{v\}$, or
- $Ant_2 \vdash Ant_1, Con_1 \vdash Con_2, S_1 \in \{c\}, S_2 \in \{c\}$ and $t_{c2} \geq t_{c1}$, or
- $Ant_1 \vdash Con_2, S_1 \in \{a, v\}$ and $t_{c2} \geq t_{a1}$.

Axiom 4 compares two commitments with each other. The antecedent in each commitment can be $\top$. Here the general idea is that, one commitment is more beneficial than a second one from the perspective of the creditor if the creditor gets more by doing less. This clause is formalized by $Ant_2 \vdash Ant_1$, $Con_1 \vdash Con_2$. This is similar to the strength relation as proposed in (Chopra and Singh, 2009). Moreover, the deadline comparisons are compatible with (Kafalı et al., 2010; Kafalı and Torroni, 2012). Among two commitments, if their states are identical or a state is more beneficial than a second state (Definition 7), then the commitment is as-good-as the second commitment. For example, $C^f(funds, refurbished[25]) \Vdash C^f(\top, refurbished[30])$ (denoted with nodes 7 and 3 in Figure 5). Here both commitments are fulfilled with the same consequent and the first commitment has an earlier deadline. Moreover, $C^f(funds, refurbished[25])$ $\Vdash C^a(funds, refurbished[30])$. Here, the fulfilled commitment entails $refurbished$, thus is as-good-as the active commitment towards $refurbished$.

Theorem 1. $\Vdash$-relation is a preorder, i.e., it is reflexive, non-symmetric, and transitive.

Our as-good-as relation is a preorder. The proof can be found in the Appendix. The

above development shows how a term is as-good-as another term. Starting from that, we need to show how a state is as-good-as another state. As a motivating example, consider the states in Figure 6. To decide whether the top state is as-good-as the bottom state, we need to figure out whether there exists a conjunction of terms on the top state for each term on the bottom state such that the conjunction of terms on the top is as good as the term on the bottom. That is, the term $funds$ in state $\mathcal{S}^{25}(bob)$ is as-good-as the term $funds$ in state $\mathcal{S}^{25}(bob_p)$. Similarly, the term $\mathcal{C}^{f}_{cnt,bob}(\top, refurbished[30])$ is as good as the term $refurbished$.

Definition 9. State $\mathcal{S}^T(X) = \langle \Phi, \mathcal{C} \rangle$ is as-good-as term $\mathcal{T}_j$, denoted $\mathcal{S}^T(X) \Vdash \mathcal{T}_j$, iff $\exists$ $\alpha = \mathcal{T}_n \wedge ... \mathcal{T}_i ... \wedge \mathcal{T}_m$: for each $\mathcal{T}_i \in \alpha$, $\mathcal{T}_i \in \Phi \cup \mathcal{C}$ and $\alpha \Vdash \mathcal{T}_j$.

According to Definition 9, a state is as-good-as a term if there is a conjunction of terms in the state that is as-good-as the term.

Definition 10. State $\mathcal{S}^{T_m}(X)$ is as-good-as state $\mathcal{S}^{T_n}(Y)$, denoted $\mathcal{S}^{T_m}(X) \Vdash \mathcal{S}^{T_n}(Y)$, iff $\forall\, \mathcal{T} \in \mathcal{S}^{T_n}(Y)$: $\mathcal{S}^{T_m}(X) \Vdash \mathcal{T}$.

According to Definition 10, a state is as-good-as another state if the former state is as-good-as every term in the latter state. By computing the as-good-as relation as explained above, an agent can identify a problem and take an action as it sees fit.

Theorem 2. For an agent A, at time T, given a proposition Con, $\mathcal{S}^T(A) \nVdash \mathcal{S}^T(A_p)$ iff $takeAction(A, T, Con)$.

Our as-good-as relation is sound and complete. That is, it identifies that the agent's current state is not as-good-as its expectation only when there is a problem (soundness), e.g., the agent needs to take an action, and if there is a problem it always identifies that the agent's current state is not as-good-as its expectation (completeness). The proof can be found in the Appendix.

Finally, the agent can check its progress at any time using the state comparison described in Definition 10. This is explained in Definition 11.

Definition 11. Agent A checks whether $\mathcal{S}^T(A)$ is as-good-as $\mathcal{S}^T(A_p)$ with $checkProgress(A, T)$=Φ_{unsat}. If $\mathcal{S}^T(A)$ is not as-good-as $\mathcal{S}^T(A_p)$, then Φ_{unsat} contains the terms that are not satisfied in $\mathcal{S}^T(A_p)$.

5. Operational Reasoning in PISAGOR

We now build the operational rules for PISAGOR to follow, that make use of the as-good-as relation. We group these rules in three different categories:

Manage expectations: Based on the creation of new commitments, the agent should update its expectations about the subject commitment's propositions. The agent can create or update its expectations following the below rules.

Create initial expectation: When a new commitment is created, the agents participating in the commitment create expectations about the antecedent and the consequent if no such expectations exist before.

$$\frac{create(C^c(Ant[t_a], Con[t_c])) \wedge \nexists \mathcal{S}^t(p) = \langle \Phi, \mathcal{C} \rangle : Ant \in \Phi}{create(\mathcal{S}^t(p) = \langle \{Ant\}, \mathcal{C} \rangle)}$$

$$\frac{create(C^c(Ant[t_a], Con[t_c])) \wedge \nexists \mathcal{S}^t(p) = \langle \Phi, \mathcal{C} \rangle : Con \in \Phi}{create(\mathcal{S}^t(p) = \langle \{Con\}, \mathcal{C} \rangle)}$$

Update existing expectation: When a new commitment is created and a participating agent already has expectations about the antecedent or the consequent, then it will only update its expectation if the new deadline proposed by the commitment is earlier than the existing one.

$$\frac{create(C^c(Ant[t_a], Con[t_c])) \wedge \exists \mathcal{S}^t(p) = \langle \Phi, \mathcal{C} \rangle : Ant \in \Phi \wedge t_a < t}{remove(\mathcal{S}^t(p) = \langle \{Ant\}, \mathcal{C} \rangle) \wedge create(\mathcal{S}^{t_a}(p) = \langle \{Ant\}, \mathcal{C} \rangle)}$$

$$\frac{create(C^c(Ant[t_a], Con[t_c])) \wedge \exists \mathcal{S}^t(p) = \langle \Phi, \mathcal{C} \rangle : Con \in \Phi \wedge t_c < t}{remove(\mathcal{S}^t(p) = \langle \{Con\}, \mathcal{C} \rangle) \wedge create(\mathcal{S}^{t_c}(p) = \langle \{Con\}, \mathcal{C} \rangle)}$$

Check progress: When the time reaches a point where the agent has an expectation, then the agent has to check its current progress against the expectation using the as-good-as relation as follows:

$$\frac{currentTime = t \wedge \exists \mathcal{S}^t(A_p) = \langle \Phi, \mathcal{C} \rangle : \Phi \cup \mathcal{C} \neq \emptyset}{checkProgress(A, t) = \Phi_{unsat}}$$

Take action: If the current state is not as-good-as the expectation, then the agent has to take appropriate action as follows, e.g., create a new commitment about the unsatisfied proposition:

$$\frac{checkProgress(A, t) = \Phi_{unsat} \wedge \exists \phi \in \Phi_{unsat}}{create(C^c(Ant[t], \phi[t]))}$$

Next, we demonstrate how these rules can be operated by the agent to manage its progress throughout the protocol, via a case study based on Example 1.

6. Implementation & Case Study

We have implemented PISAGOR using the Reactive Event Calculus ($\mathcal{REC}$). $\mathcal{REC}$ has been developed to monitor commitments in run-time (Chesani et al., 2013; Kafalı et al., 2010; Wood et al., 2013), and it is a logic used for representing events and their outcomes. While the Event Calculus, in its nature, is used for backward reasoning (e.g., goal-driven), $\mathcal{REC}$ enables forward reasoning (e.g., event-driven) as well. We employ $\mathcal{REC}$ as the reasoner component of the agent shown in Figure 1. Note that each agent has a separate $\mathcal{REC}$ engine that it can run at any time throughout the execution. The $\mathcal{REC}$ engine takes three types of input:

- *Commitment theory* contains the rules on how commitments are manipulated. This is a shared rule-base for all the agents.
- *Protocol rules* contains the domains rules that describe the consequences of the agents' actions as well as other domain facts. This is an agent-dependent as well as domain-dependent rule-base as each agent has a separate protocol description that covers its own view.

– *Event trace* contains the actions that the agents perform throughout time. Like the protocol description, the event trace is also agent-dependent. That is, each agent is aware of the events that are related to it, but does not see the events that might take place among other agents.

```
% create as conditional
initiates(E, status(C, conditional), T):-
        ccreate(E, C, T).

% conditional to active
terminates(E, status(C, conditional), T):-
        detach(E, C, T).

initiates(E, status(C, active), T):-
        detach(E, C, T).

detach(E, c(X, Y, Ant, Ta, Con, Tc), T):-
        conditional(c(X, Y, Ant, Ta, Con, Tc), T),
        T =< Ta,
        initiates(E, Ant, T).

...

% active to violated
terminates(E, status(C, active), T):-
        violate(E, C, T).

initiates(E, status(C, violated), T):-
        violate(E, C, T).

violate(E, c(X, Y, Ant, Ta, Con, Tc), T):-
        active(c(X, Y, Ant, Ta, Con, Tc), T),
        T > Tc.
```

Listing 1: Commitment theory in $\mathcal{REC}$.

```
checkProgress(Term, satisfied, T):-
        asGoodas(Term2, Term, T).

% Axiom 1 (proposition is as-good-as proposition)
asGoodas(Con, proposition(Con), T):-
        holds_at(Con, T).

% Axiom 2 (proposition is as-good-as commitment)
asGoodas(Con, commitment(c(X, Y, Ant, Ta, Con, Tc), State), T):-
        holds_at(Con, T),
        T =< Tc.

% Axiom 3 (commitment is as-good-as proposition)
asGoodas(c(X, Y, Ant, Ta, Con, Tc), proposition(Ant), T):-
        active(c(X, Y, Ant, Ta, Con, Tc), T).
...

% Axiom 4 (commitment is as-good-as commitment)
asGoodas(c(X1, Y1, Ant, Ta1, Con, Tc1),
        commitment(c(X2, Y2, Ant, Ta2, Con, Tc2), State), T):-
        fulfilled(c(X, Y, Ant, Ta1, Con, Tc1), T),
        Tc1 =< Tc2.
...
```

Listing 2: Formulating as-good-as in $\mathcal{REC}$.

Once the $\mathcal{REC}$ engine is run with above input, it produces an outcome that demonstrates the fluents the agent is aware of through time (e.g., states of commitments). This

process is often called commitment tracking (Chesani et al., 2013). Listing 1 describes in part how the commitment theory is implemented in $\mathcal{REC}$. A commitment can be created as conditional (lines 2-3). That is, when an event happens that conditionally creates a commitment, then the state of the commitment becomes conditional active. This is done through the domain-independent EC predicate *initiates/3*. Similarly, when an event happens that detaches the commitment, then the state of the commitment is no longer conditional (lines 6-7) and it becomes active (lines 9-10). A commitment is detached when its antecedent is initiated before the deadline Ta (lines 12-15). When the commitment is active and its consequent is not initiated before the deadline Tc, then the commitment is violated (lines 26-28).

In this paper, we do not use $\mathcal{REC}$ only for monitoring commitment states, but also to identify problems at run-time. Thus, we further integrate the agent's expectations as well as the as-good-as relation for determining when to take actions. Listing 2 describes in part how the as-good-as relation is implemented in $\mathcal{REC}$. When checking for the agent's progress, if there is a term in the agent's current state that is as-good-as the term in the expected state, then its expectation is satisfied (lines 1-2). The rest of the listing develops this theory for the four axioms described in Section 4. An expected proposition is satisfied if the same proposition holds in the current state at that time (lines 5-6). Similarly, an expected commitment is satisfied if the consequent of the commitment holds in the current state T before the deadline Tc. Note that we only give the first rules for Axioms 3 and 4. Other rules are described similarly.

Listing 3 describes in part how the operational rules are implemented in $\mathcal{REC}$. When a new commitment is created, the agent creates an expectation about its antecedent if no such expectation holds before (lines 2-4). If such an expectation already holds, then the agent only updates it if an earlier deadline is proposed by the new commitment, i.e., removes the previous expectation (lines 8-11) and creates a new expectation (lines 13-16). The rules for the consequent are described similarly. The agent takes an action if it finds out that an expectation is not satisfied after checking the progress (lines 20-22).

```
1  % create expectation when a commitment is created
2  initiates(E, expectation(proposition(Con), Tc), T):-
3          ccreate(E, c(X, Y, Ant, Ta, Con, Tc), T),
4          \+ holds_at(expectation(proposition(Con), _), T).
5  ...

7  % update expectation if an earlier deadline is proposed
8  terminates(E, expectation(proposition(Ant), Te), T):-
9          ccreate(E, c(X, Y, Ant, Ta, Con, Tc), T),
10         holds_at(expectation(proposition(Ant), Te), T),
11         Ta < Te.

13 initiates(E, expectation(proposition(Ant), Ta), T):-
14         ccreate(E, c(X, Y, Ant, Ta, Con, Tc), T),
15         holds_at(expectation(proposition(Ant), Te), T),
16         Ta < Te.
17 ...

19 % check progress and take action
20 initiates(E, takeAction(Term), T):-
21         holds_at(expectation(Term, T), T),
22         \+ checkProgress(Term, satisfied, T).
```

Listing 3: Operational rules in $\mathcal{REC}$.

Listing 4 describes how the domain model of Bob is implemented in $\mathcal{REC}$. Actions of the agents initiate the corresponding fluents (lines 1-3). The offer from the estate

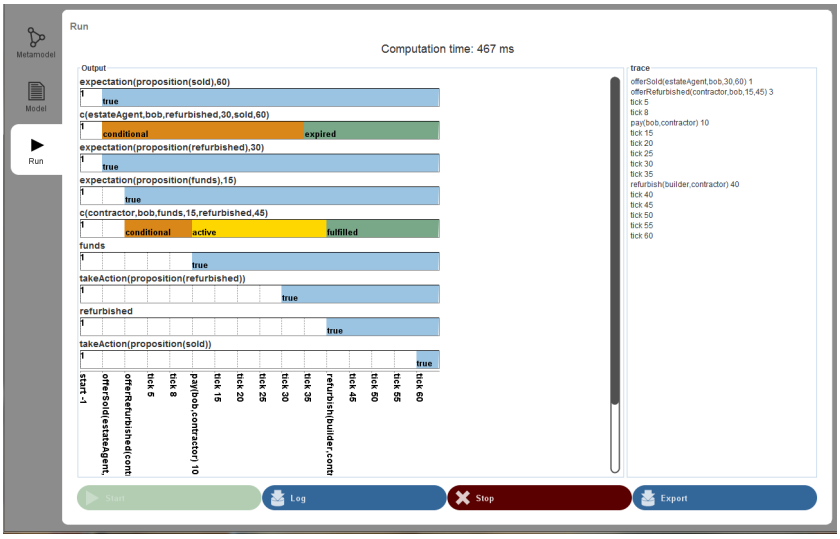


Fig. 7. $\mathcal{REC}$ interface for the output of Case I. It is made up of three tabs (shown on the left): (i) *metamodel* includes the commitment theory and as-good-as axioms, (ii) *model* describes the domain model of the agent, and (iii) *run* enables execution given a trace of events that correspond to the agent's actions (shown on the right). The plot in the middle shows the progression of fluents through time (i.e., according to the effects of agent's actions).

agent to Bob creates the conditional commitment between them: if Bob refurbishes the house by the deadline Ta, then the estate agent sells the house by the deadline Tc (lines 5-6). Similarly, the offer from the contractor to Bob creates the conditional commitment between them (lines 7-8). The complete $\mathcal{REC}$ implementation can be found online[1].

```
1 initiates(exec(pay(bob, contractor)), funds, T).
2 initiates(exec(refurbish(builder, contractor)), refurbished, T).
3 initiates(exec(sell(estateAgent, bob)), sold, T).

5 ccreate(exec(offerSold(estateAgent, bob, Ta, Tc)),
6         c(estateAgent, bob, refurbished, Ta, sold, Tc), T).
7 ccreate(exec(offerRefurbished(contractor, bob, Ta, Tc)),
8         c(contractor, bob, funds, Ta, refurbished, Tc), T).
```

Listing 4: Domain model for Bob in $\mathcal{REC}$.

Let us now review two cases following Example 1 to demonstrate the usage of the as-good-as relation for determining when PISAGOR needs to take action.

Case I: Consider the following trace of events:

```
 1 offerSold(estateAgent,bob,30,60)
 3 offerRefurbished(contractor,bob,15,45)
10 pay(bob,contractor)
40 refurbish(builder,contractor)
```

[1] http://mas.cmpe.boun.edu.tr/ozgur/code.html

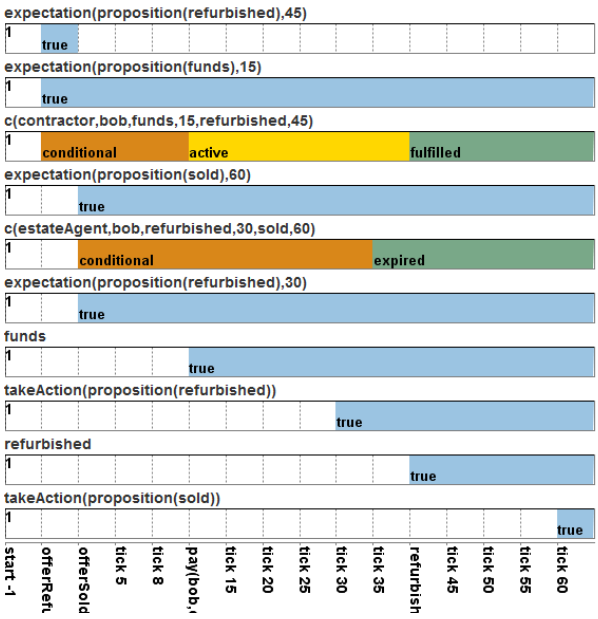


Fig. 8. $\mathcal{REC}$ output for Case II.

At time 1, the estate agent offers Bob to sell his house by time 60 if Bob refurbishes the house by time 30. Then, at time 3, the contractor offers Bob to refurbish his house by time 45 if Bob sends the payment by time 15. Accordingly, Bob pays at time 10 and the contractor gets the house refurbished at time 40. Figure 7 shows the $\mathcal{REC}$ interface that corresponds to this case. When run, $\mathcal{REC}$ produces the figure shown in the middle. The horizontal axis shows the timeline of events that happened during Bob's execution. Notice a *tick* event is associated with some non-occupied (i.e., no protocol events) discrete time points. This is required for $\mathcal{REC}$ to process properly since it is event-driven, e.g., a new event triggers $\mathcal{REC}$ to process further. The fluents are positioned vertically and their truth values (and the corresponding states for commitments) are computed according to the events.

When the estate agent makes the offer, Bob creates expectations about the two propositions: $refurbished$ for time 30 and $sold$ for time 60. Then after the offer from the contractor, Bob creates another expectation about proposition $funds$ for time 15. Note that Bob does not update its expectation about $refurbished$ for 45 since he already has the expectation for an earlier deadline. When the time is 15, Bob checks his progress and sees that proposition $funds$ already holds. Thus, there is no need to take an action about it. When the time is 30, Bob again checks his progress and this time PISAGOR identifies a problem, i.e., the fluent $takeAction(proposition(refurbished))$ becomes true. Note that the contractor's commitment is not violated. However, Bob needs to take action, e.g., create a new commitment to get a quicker refurbishment for the house.

Case II: Consider the following trace of events:

```
1 offerRefurbished(contractor,bob,15,45)
3 offerSold(estateAgent,bob,30,60)
```

```
10 pay(bob,contractor)
40 refurbish(builder,contractor)
```

Notice that the offers are reversed from the first case. The rest of the trace is the same. Figure 8 shows the $\mathcal{REC}$ output that corresponds to this case. Bob creates an expectation about $refurbished$ for time 45 initially. But, when the second commitment is created, Bob updates this expectation for time 30. When the time is 30, although the contractor's commitment is not violated, again Bob needs to take an action about the refurbishment of the house since his deadline for the estate agent's commitment has come.

7. Discussion

In this section, we first compare our work with relevant literature. Then, we review our contributions and provide directions for possible extensions.

Related Work: Agents have been utilized to support users in e-commerce, providing content support, interface support and decision support for making optimal purchasing decisions (Menczer, Street, Vishwakarma, Monge and Jakobsson, 2002; Chang and Wong, 2011). These approaches offer search and choice behaviors for different purchasing alternatives. In (Shimazu, 2002), the authors build automated salesclerks based on real conversations of humans to propose sample products by observing customer responses. The idea of artificial shopping agents is discussed to maximize benefits, minimize time and effort, and determine possible concerns. While these directions help improve the customer experience in e-commerce markets, they do not provide support for the customers when unexpected situations occur, e.g., a late delivery. We, on the other hand, provide a systematic way to detect such cases based on the expectations of users. Privacy of users and fraud detection are other concerns in e-commerce (Berendt, Günther and Spiekermann, 2005; Huang et al., 2012). In (Berendt et al., 2005), the authors compare users' privacy concerns with their actual behaviors via monitoring and controlling their actions. In a sense, we take a similar approach, and compare the users' expectations with the actual events happened in their environment to see whether they are progressing as expected (e.g., catch abnormal interactions in the case of frauds).

Detecting problematic situations is of utmost importance in many computational environments including distributed systems and multiagent systems. This presents itself especially when autonomous entities need to interact with each other (Moubaiddin and Obeid, 2013; Nguyen and Katarzyniak, 2009). When software agents enact business protocols together, problems might take place and diagnosing their cause allows entities to deal with the problem in a timely manner. Hence, instead of a failed execution, the entities, either individually or cooperatively, can work to solve the problem. In this work, we focus on a subset of e-commerce protocols where the interactions among participants are governed by contracts. Contracts and norms have been extensively studied to regulate agent interaction also in other domains (Meneguzzi, Modgil, Oren, Miles, Luck and Faci, 2012). The monitoring and verification of contracts has been covered before in the literature (El Menshawy, Benthar, Qu and Dssouli, 2011; Giannikis and Daskalopulu, 2011). However, these works take a centralized perspective where reasoning on contracts are performed by a single entity that has a global view of the environment. Moreover, since monitoring is done centrally, they do not provide subjective reasoning as we do here, e.g., a problem situation for a participant might be considered normal by another participant based on their expectations from the contract.

This type of reasoning would support participants with their decision making (Huang and Kuo, 2012).

In central monitoring systems, tracking the states of individual commitments is an effective way to detect problems, since all the interactions of agents are observable. Such tracking of commitments can be done with $\mathcal{REC}$ (Chesani et al., 2013). In this paper, we are interested in monitoring interactions when agents have local views of the environment and their ideas of problematic situations vary. Thus, we do not take a centralized approach. Instead, we need to treat multiagent systems as embodying interaction protocols that are only partially shared and analyze problems as cases raising through discrepancies in agent's expectations of the system. Another approach that uses the Event Calculus is described in (Farrell, Sergot, Sall and Bartolini, 2005). The authors use an XML-based formalisation called ecXML, and a GUI-based program for visually tracking the states of contracts. Moreover, we do not have the aim of analyzing the system at design time in the case of model checkers (El Menshawy et al., 2011; Telang and Singh, 2012*b*; El Kholy, El Menshawy, Bentahar, Qu and Dssouli, 2013). While this is certainly useful, many times certain problems may not be detected at design time. Hence, our development in this paper is towards creating methods to determine when agents need to take actions at run time.

Commitments account for the constitutive specification of a protocol. They provide flexible execution for the agents as long as their commitments are satisfied. When a problem occurs (e.g., a commitment is violated), there may be several reasons behind it. One such reason is the misalignment of agents' commitments. That is, the debtor and the creditor have different understandings for the same commitment (Chopra and Singh, 2009; Kafalı et al., 2010). In order for agents to interoperate correctly in a distributed protocol, their commitments should be aligned. For example, if Bob expects delivery today, and the seller thinks the payment is not processed yet (assume payment is required prior to delivery), then there is obviously a problem between the two agents. In this work, we do not consider problems caused by misalignment, e.g. different observations of the agents. Rather, we focus on the expectations of agents, and the problems that stem from the differences of their expectations.

Rittgen discusses two types of contracts: behavior-based and outcome-based (Rittgen, 2008). While the author claims that an outcome-based approach should be favored since it might be costly to monitor agents, a contract-based architecture for behavior-based contracts is given for interorganizational processes. In this sense, our approach can be considered outcome-based as we do not focus on the behavior of the agents as long as their commitments are fulfilled. However, by taking into account expectations, we can also represent expected behaviors from agents.

Other key properties of multi-party business processes are the conformance and interoperability of agents. First, the agent's design should conform to the protocol's standards (Chopra et al., 2010; Telang and Singh, 2012*a*). In other words, the agent should have the necessary capabilities to satisfy its commitments. Moreover, the agent should be interoperable with other agents. That is, all agents should infer the same semantics for the protocol rules. Consider the following intuitive example: Bob wishes to pay using his credit card, however the seller only accepts bank transfer as payment. Obviously, these two agents are not interoperable, since payment by credit card is not defined in the seller's rule-base. Assume they have a commitment which tells if payment is processed, then delivery will be done. Now, when Bob pays by credit card, he thinks that the seller will deliver. However, looking from the seller's point of view, the seller is not committed to deliver since the payment is not processed via a bank transfer. Here, we assume the agents both conform to the protocol they are executing and are interoperable with each other. Even then, problems might occur. We investigate such cases.

Governatori discusses the compliance of business processes to a normative system (Governatori, 2013). They model business processes with the standard BPMN notation, in which tasks can be related to each other via different relations, e.g., xor. A normative system is described in terms of a set of different obligation types. Firstly, they differentiate between punctual (that are in force for a given task) and persistent obligations. In our framework, all commitments are persistent as we do not associate the lifecycle of a commitment with a task's duration. Moreover, we only support achievement type commitments in this work. Maintenance commitments is left for future work. When there is a single conditional commitment, this corresponds to a preemptive obligation. Non-preemptive obligations can also be modelled using another base level commitment for the antecedent, which forces an obligation on the antecedent as well. In addition, compensable obligations can be represented by having another conditional commitment with the antecedent being the negation of the former commitment's consequent and the consequent being the proposition for the actual compensation. When checking for compliance, they assume that the obligations are already in force as well as the tasks that make up the process. We also propose operational rules to describe how an agent's expectations can be established based on the created commitments. Moreover, in (Governatori and Milosevic, 2006) the authors propose an XML-based contract language to reason about violations and their possible compensations.

There has been some work in the literature on comparing agents' states as we do in this work. One related work is that of Mallya and Singh (Mallya and Singh, 2007), which focuses on similarity relations for protocol runs. In parallel with our work, a state is described as a set of propositions and commitments, and several similarity relations are given to compare states. The idea is to compare protocol runs in terms of states. One major difference between their similarity relations and our *as-good-as* relation is that their relations are equivalence relations (e.g., supports symmetry) while ours is not. This is mainly related to the motivation behind those relations. That is, they aim to merge smaller protocols into larger ones by comparing protocol states. We, on the other hand, provide a one-way relation (e.g., not necessarily symmetric) to compare states. Since our main motivation is to ensure that an agent's state complies with its expectations, the inverse direction is not significant at all (e.g., whether expectations support actual states). Moreover, it is not necessary to ensure that both states are equivalent.

Contributions: In this paper, we have proposed the as-good-as relation that can be used to compare agents' states. We have implemented PISAGOR: a software agent that realizes this relation to tell whether there is a problem and that the user needs to take an action. That is, if the agent's current state is not as-good-as its expected state, then the agent identifies a problem. Moreover, we proposed operational rules so that PISAGOR can create expectations based on its commitments.

We extend the concept of problematic situations that are described in the literature. Often, a problem is considered identical to a contract violation. While we accommodate that perspective, we also take into account the agent's expectations from their commitments. Note that we would normally identify a problem when a commitment is still active when it should be fulfilled. However, by letting the expected state to be constructed according to the agent's expectations, a significant problem is avoided.

Future work: For this work, we have focused on commitments where the proposition needs to be achieved once for the commitment to be fulfilled. It is also interesting to capture cases where the proposition needs to be maintained over a period of time, e.g., the contractor committing to Bob for the house staying in good condition for two years after the refurbishment. We plan to investigate the relations among maintenance type commitments in the future.

The as-good-as relation can also be utilized to perform prediction of problems. One way to do so is to compare the expected states of agents. For example, assume that the contractor expects delivery of materials today. However, the supplier has additional information about an accident that the contractor is not aware of, and does not foresee delivery any time soon. Now, if the contractor can compare its expectation with that of the supplier, then he may predict that there will be a problem. Such reasoning would require parties to share their expectations and other relevant information, e.g., the contractor reviews the website of the supplier for delivery times. This is another direction that we would like to pursue in a future work.

Acknowledgements. We thank the anonymous reviewers for their useful comments and suggestions. Most of this work was done while Özgür Kafalı was doing research in Bogazici University and Royal Holloway, University of London. This research is supported by Bogazici University Research Fund under grant BAP13A01P2 and the Scientific and Technological Research Council of Turkey (TUBITAK) under grant 113E543.

Appendix

Theorem 1 $\Vdash$-relation is a preorder, i.e., it is reflexive, non-symmetric, and transitive.

Proof. It is trivial that $\Vdash$is reflexive and non-symmetric. For atomic propositions, reflexivity follows from Axiom 1 and for commitments it follows from Axiom 4. For showing non-symmetry, any unidirectional edge in Figure 5 serves as a counter-example for symmetry. Consider the edge between nodes 2 and 5. The proposition $refurbished$ satisfies the violated base-level commitment $C^v(\top, refurbished)$, but the other direction is not true. The edges between nodes 6–7 and 1–6 show similar properties.

For transitivity, we need to show that if $\alpha \Vdash \beta$ and $\beta \Vdash \gamma$ then $\alpha \Vdash \gamma$. Assume α is a proposition Con, β and γ are both base-level commitments with consequents Con. By Axiom 2, $\alpha \Vdash \beta$, independently of the state of β. Now, when (1) the state of β is fulfilled, by Axiom 4, $\beta \Vdash \gamma$, independently of the state of γ. Then, $\alpha \Vdash \gamma$, by Axiom 2. When (2) the state of β is active, by Axiom 4, $\beta \Vdash \gamma$, if the state of γ is active or violated. Then, again $\alpha \Vdash \gamma$, by Axiom 2. When (3) the state of β is violated, by Axiom 4, $\beta \Vdash \gamma$, if the state of γ is also violated. Then, again $\alpha \Vdash \gamma$, by Axiom 2. Thus, $\Vdash$is transitive. Other combinations of α, β and γ follow similarly. □

Theorem 2 For an agent A, at time T, given a proposition Con, $\mathcal{S}^T(A) \nVdash \mathcal{S}^T(A_p)$ iff $takeAction(A, T, Con)$.

Proof. We prove both directions by contradiction.

I. Forward direction (Completeness): If $takeAction(A, T, Con)$, then $\mathcal{S}^T(A) \nVdash \mathcal{S}^T(A_p)$.

Let $\exists Con$: $takeAction(A, T, Con)$. Then, by Definition 8, either $\mathcal{T}_{Con}(\mathcal{S}^T(A)) = \emptyset$, or $\exists\, x$: $x \in \mathcal{T}_{Con}(\mathcal{S}^T(A))$. First, consider the former case. Assume $\mathcal{S}^T(A) \Vdash \mathcal{S}^T(A_p)$. This will occur only if $\mathcal{T}_{Con}(\mathcal{S}^T(A_p)) = \emptyset$ also. Otherwise, a term that includes Con cannot be satisfied according to the satisfiability axioms. This is a contradiction, because, by Definition 8, $\mathcal{T}_{Con}(\mathcal{S}^T(A_p))$ cannot be empty.

Now, consider the latter case. Again, assume $\mathcal{S}^T(A) \Vdash \mathcal{S}^T(A_p)$. There are three possibilities:

1. $\mathcal{S}^T(A_p) = \langle \emptyset, \emptyset \rangle$. Then, every state is as-good-as the empty state since there are no terms in it to satisfy (Definition 10). This is a contradiction, because, by Definition 8, $\mathcal{T}_{Con}(\mathcal{S}^T(A_p))$ cannot be empty.

2. $\mathcal{T}_{Con}(\mathcal{S}^T(A_p)) = \emptyset$. Then, there is no need to satisfy a term that includes Con. This is a contradiction, because, again by Definition 8, $\mathcal{T}_{Con}(\mathcal{S}^T(A_p))$ cannot be empty.
3. $\exists\, y$: $y \in \mathcal{T}_{Con}(\mathcal{S}^T(A_p))$. Then, x is as-good-as y (since both are the terms that include Con). There are three possibilities:
   (a) $x = Con$ or $x = C^f(Con)$. Then, according to the satisfiability axioms, $x \Vdash\!\!\vdash y$ for any y. This is a contradiction, because, by Definition 8, $u(x)$ cannot be greater than or equal to $u(y)$.
   (b) $x = C^a(Con)$ and ($y = C^a(Con)$ or $y = C^v(Con)$). Then, by Axiom 4, $x \Vdash\!\!\vdash y$. This is a contradiction, because, by Definition 8, $u(x)$ cannot be greater than or equal to $u(y)$.
   (c) $x = C^v(Con)$ and $y = C^v(Con)$. Then, by Axiom 4, $x \Vdash\!\!\vdash y$. This is a contradiction, because, by Definition 8, $u(x)$ cannot be equal to $u(y)$.

Hence, we conclude that whenever the agent takes an action, $\mathcal{S}^T(A) \not\Vdash\!\!\vdash \mathcal{S}^T(A_p)$.

II. Backward direction (Soundness): If $\mathcal{S}^T(A) \not\Vdash\!\!\vdash \mathcal{S}^T(A_p)$, then $\exists Con$: $takeAction(A, T, Con)$.

Let $\mathcal{S}^T(A) \not\Vdash\!\!\vdash \mathcal{S}^T(A_p)$. Then, by Definition 10, $\exists\, y$: $y \in \mathcal{S}^T(A_p)$ and $\mathcal{S}^T(A) \not\Vdash\!\!\vdash y$. That is, there is at least one term that is not satisfiable. Let $y \in \mathcal{T}_{Con}(\mathcal{S}^T(A_p))$. That is, Con be the proposition related to the term y (either y itself or the consequent of y in case y is a commitment). Now, assume $takeAction(A, T, Con)$ does not hold. Then, $\exists$ x: $x \in \mathcal{S}^T(A)$ and $u(x) \geq u(y)$. Otherwise, $takeAction(A, T, Con)$ would hold. There are three possibilities:

1. $x = Con$ or $x = C^f(Con)$. Then, $u(x) \geq u(y)$ for any y, and no action needs to be taken. This is a contradiction, because, according to the satisfiability axioms x would be as-good-as any such y.
2. $x = C^a(Con)$ and ($y = C^a(Con)$ or $y = C^v(Con)$). Then, $u(x) \geq u(y)$, and no action needs to be taken. This is a contradiction, because, by Axiom 4, x would be as-good-as y.
3. $x = C^v(Con)$ and $y = C^v(Con)$. Then, $u(x) = u(y)$, and no action needs to be taken. This is a contradiction, because, by Axiom 4, x would be as-good-as y.

□

References

Berendt, B., Günther, O. and Spiekermann, S. (2005), 'Privacy in e-commerce: stated preferences vs. actual behavior', *Commun. ACM* **48**(4), 101–106.

Chang, J.-S. and Wong, H.-J. (2011), 'Selecting appropriate sellers in online auctions through a multi-attribute reputation calculation method.', *Electronic Commerce Research and Applications* **10**(2), 144–154.

Chesani, F., Mello, P., Montali, M. and Torroni, P. (2013), 'Representing and monitoring social commitments using the event calculus', *Autonomous Agents and Multi-Agent Systems* **27**(1), 85–130.

Chopra, A. K., Dalpiaz, F., Giorgini, P. and Mylopoulos, J. (2010), Reasoning about agents and protocols via goals and commitments, *in* 'Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)', pp. 457–464.

Chopra, A. K. and Singh, M. P. (2009), Multiagent commitment alignment, *in* 'Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2', AAMAS '09, pp. 937–944.

El Kholy, W., El Menshawy, M., Bentahar, J., Qu, H. and Dssouli, R. (2013), Representing and reasoning about communicative conditional commitments, *in* 'Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems', pp. 1169–1170.

El Menshawy, M., Benthar, J., Qu, H. and Dssouli, R. (2011), On the verification of social commitments and time, *in* 'Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)', pp. 483–490.

Farrell, A. D. H., Sergot, M. J., Sall, M. and Bartolini, C. (2005), 'Using the event calculus for tracking the normative state of contracts.', *Int. J. Cooperative Inf. Syst.* **14**(2-3), 99–129.

Giannikis, G. K. and Daskalopulu, A. (2011), 'Normative conflicts in electronic contracts', *Electron. Commer. Rec. Appl.* **10**(2), 247–267.

Governatori, G. (2013), 'Business process compliance: An abstract normative framework', *it - Information Technology* **55**(6), 231–238.

Governatori, G. and Milosevic, Z. (2006), 'A formal analysis of a business contract language.', *Int. J. Cooperative Inf. Syst.* **15**(4), 659–685.

Gupta, S. and Chakraborti, S. (2013), Utilsim: Iteratively helping users discover their preferences, *in* C. Huemer and P. Lops, eds, 'E-Commerce and Web Technologies', Vol. 152 of *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, pp. 113–124.

Huang, S.-M., Hua, J.-S., Will, H. and Wu, J.-W. (2012), 'Metamodeling to control and audit e-commerce web applications', *Int. J. Electron. Commerce* **17**(1), 83–118.

Huang, Y.-F. and Kuo, F.-Y. (2012), 'How impulsivity affects consumer decision-making in e-commerce', *Electron. Commer. Rec. Appl.* **11**(6), 582–590.

Jakob, M., Pěchouček, M., Miles, S. and Luck, M. (2008), Case studies for contract-based systems, *in* 'Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)', pp. 55–62.

Kafalı, Ö., Chesani, F. and Torroni, P. (2010), What happened to my commitment? Exception diagnosis among misalignment and misbehavior, *in* 'Computational Logic in Multi-Agent Systems', Vol. 6245 of *Lecture Notes in Computer Science*, pp. 82–98.

Kafalı, Ö. and Torroni, P. (2012), 'Exception diagnosis in multiagent contract executions', *Annals of Mathematics and Artificial Intelligence* **64**(1), 73–107.

Kafalı, Ö. and Yolum, P. (2011), A distributed treatment of exceptions in multiagent contracts, *in* 'Proceedings of the 9th International Workshop on Declarative Agent Languages and Technologies (DALT)'.

Kersten, G. E., Vahidov, R. M. and Gimon, D. (2013), 'Concession-making in multi-attribute auctions and multi-bilateral negotiations: Theory and experiments', *Electronic Commerce Research and Applications* **12**(3), 166–180.

Mallya, A. U. and Singh, M. P. (2007), 'An algebra for commitment protocols', *Autonomous Agents and Multi-Agent Systems* **14**(2), 143–163.

Menczer, F., Street, W. N., Vishwakarma, N., Monge, A. E. and Jakobsson, M. (2002), Intellishopper: a proactive, personal, private shopping assistant, *in* 'Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)', pp. 1001–1008.

Meneguzzi, F., Modgil, S., Oren, N., Miles, S., Luck, M. and Faci, N. (2012), 'Applying electronic contracting to the aerospace aftercare domain', *Eng. Appl. Artif. Intell.* **25**(7), 1471–1487.

Morid, M. A. and Shajari, M. (2012), 'An enhanced e-commerce trust model for community based centralized systems.', *Electronic Commerce Research* **12**(4), 409–427.

Moubaiddin, A. and Obeid, N. (2013), 'On formalizing social commitments in dialogue and argumentation models using temporal defeasible logic.', *Knowl. Inf. Syst.* **37**(2), 417–452.

Nguyen, N. T. and Katarzyniak, R. (2009), 'Actions and social interactions in multi-agent systems', *Knowl. Inf. Syst.* **18**(2), 133–136.

Özer, A. H. and Özturan, C. (2011), 'Multi-unit differential auction-barter model for electronic marketplaces.', *Electronic Commerce Research and Applications* **10**(2), 132–143.

Rittgen, P. (2008), 'A contract-based architecture for business networks', *Int. J. Electron. Commerce* **12**(4), 115–146.

Shimazu, H. (2002), 'Expertclerk: A conversational case-based reasoning tool fordeveloping salesclerk agents in e-commerce webshops', *Artif. Intell. Rev.* **18**(3-4), 223–244.

Singh, M. P. and Huhns, M. N. (2005), *Service-Oriented Computing: Semantics, Processes, Agents*, John Wiley & Sons, Chichester, UK.

Telang, P., Meneguzzi, F. and Singh, M. (2013), Hierarchical planning about goals and commitments, *in* 'Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems', pp. 877–884.

Telang, P. R. and Singh, M. P. (2012*a*), Comma: a commitment-based business modeling methodology and its empirical evaluation, *in* 'Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems', pp. 1073–1080.

Telang, P. R. and Singh, M. P. (2012*b*), 'Specifying and verifying cross-organizational business models: An agent-oriented approach', *IEEE T. Services Computing* **5**(3), 305–318.

Torroni, P., Yolum, P., Singh, M. P., Alberti, M., Chesani, F., Gavanelli, M., Lamma, E. and Mello, P. (2009),

Modelling interactions via commitments and expectations, *in* 'Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models', pp. 263–284.

Wood, D., Kafalı, Ö. and Stathis, K. (2013), DIESECT: A distributed environment for simulating e-commerce contracts, *in* C. Huemer and P. Lops, eds, 'E-Commerce and Web Technologies', Vol. 152 of *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, pp. 39–50.

Yan, Z., Fong, S. and Shi, M. (2008), 'Knowledge-empowered automated negotiation system for e-commerce', *Knowl. Inf. Syst.* **17**(2), 167–191.

Yao, Y., Ruohomaa, S. and Xu, F. (2012), 'Addressing common vulnerabilities of reputation systems for electronic commerce.', *Journal of Theoretical and Applied Electronic Commerce Research* **7**(1), 1–20.

Yolum, P. and Singh, M. P. (2007), Enacting protocols by commitment concession, *in* 'Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)', pp. 116–123.

Author Biographies

Özgür Kafalı received his B.S., M.S. and Ph.D. degrees from Bogazici University, Turkey. He previously worked as a Postdoctoral Researcher from 2012 to 2014, at the Department of Computer Science, Royal Holloway, University of London, UK. He is currently a Postdoctoral Researcher at the Department of Computer Science, North Carolina State University, USA. His research interests include artificial intelligence, multiagent systems, agent communication, computational logic, e-commerce, e-health and science of security. He serves as a program committee member for the international conferences AAMAS, ECAI and PRIMA. He is also a reviewer for the international journals Autonomous Agents and Multi-Agent Systems and The Knowledge Engineering Review.

Pınar Yolum holds a Ph.D.in computer science from North Carolina State University, Raleigh, NC, USA. She is currently an Associate Professor of computer engineering in Bogazici University, Istanbul, Turkey, where she leads a research group in multiagent systems that works on trust, agent communication, and social networks. She has (co)authored more that 50 papers in selected journals and conferences. She serves on the Editorial Board of Journal of Autonomous Agents and Multiagent Systems and Journal of AI Research. She is a Board Member of the International Foundation for Autonomous Agents and Multiagent Systems. She serves as the General Co-chair of International Conference on Autonomous Agents and Multiagent Systems for 2015.

Correspondence and offprint requests to: Özgür Kafalı, Department of Computer Science, North Carolina State University, Raleigh, NC, USA. Email: rkafali@ncsu.edu