Reasoning about Agents and Protocols via Goals and Commitments

Amit K. Chopra
University of Trento, Italy
chopra@disi.unitn.it

Fabiano Dalpiaz
University of Trento, Italy
dalpiaz@disi.unitn.it

Paolo Giorgini
University of Trento, Italy
paolo.giorgini@disi.unitn.it

John Mylopoulos
University of Trento, Italy
jm@disi.unitn.it

ABSTRACT

This paper seeks to combine two largely independent threads of multiagent systems research—agent specification and protocols. We specify agents in terms of *goal models* (as used in Tropos). We specify protocols in terms of the *commitments* among agents. We illustrate and formalize the semantic relationship between agents and protocols by exploiting the relationship between goals and commitments. Given an agent specification and a protocol, the semantics helps us perform two kinds of verification: (1) whether the protocol supports achieving particular agent goals, and (2) whether the agent's specification supports the satisfaction of particular commitments.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

Keywords

Theory, Verification

General Terms

Goals, Commitments, Protocols, Communication, Agent-Oriented Software Engineering, Compliance

1. INTRODUCTION

Two strands of multiagent systems research have progressed, for the most part, independently from each other. One has to do with the specification of individual agents in terms of cognitive concepts and reasoning about strategies, and so on; the other with the specification of protocols. The two strands are related: a protocol is the specification of a multiagent system; a multiagent system is *instantiated* when specific agents adopt roles in the protocol. One of the principal challenges in multiagent systems research is to tie the specification of individual agents to protocols [8, 11, 16]. This would potentially help answer questions such as if a protocol is suitable for an agent's needs, and how an agent should behave at runtime to meet those needs. This paper seeks to address this challenge.

Agents are usefully specified in terms of cognitive concepts such as beliefs, goals, desires, intentions, obligations; these capture at a high-level the internal decision making of an agent—its policies, choices, rationale, and so on. Existing Agent-Oriented Software Engineering (AOSE) methodologies, agent programming languages, and formal accounts of agent reasoning employ cognitive concepts for agent specification but consider communication (protocols) in terms of data and control flow abstractions—for example, using notations such as UML sequence diagrams. Giving short shrift to communication means giving short shrift to the specification of multiagent systems.

Goals represent an important cognitive abstraction for agent specification. This paper adopts goal models as specified in Tropos [5] as representative of agent specifications. In contrast to existing approaches, our approach uses *commitment protocols* as a high-level abstraction for interaction [23] (referred to as simply *protocols* from now on). A protocol consists of messages along with their meanings in terms of commitments. Commitments are binding in a sociolegal way: failure to satisfy the commitment is *noncompliance* with the protocol and often results in the imposition of penalties.

The relation between goals and commitments is the following. An agent has certain goals that it wants to satisfy, and in doing so it typically must make (to others) or get (from others) commitments about certain goals. Given their autonomy, an agent cannot force another to adopt or satisfy a goal; given their heterogeneity, an agent ideally should not make assumptions about the goals of others. Thus, in general, agents can do no better than deal in commitments to achieve goals they cannot themselves achieve in isolation.

Given the above, an agent's designer (or perhaps the agent itself at runtime) would naturally want to verify if a particular protocol were suitable for the achievement of the agent's goals. He would also want to verify that if an agent makes a certain commitment, then the agent's specification is such that it supports fulfillment of the commitment. Let's consider some simple examples to illuminate our purpose.

Consider a purchase protocol with the roles merchant and customer. The protocol specifies that an offer message from the merchant to the customer means a commitment from the merchant to the customer that if the customer has paid, then the merchant will deliver the goods. Consider that Rob plays merchant. If Rob sends such a message to some customer,

Cite as: Reasoning about agents and protocols via goals and commitments, Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

then irrespective of Rob's and the customer's internal design, the commitment to the customer holds. It would be prudent to verify *a priori* that Rob can fulfill the commitment by sending the item to the customer.

Also, consider that another agent Alice has the goal of purchasing something—she plays the role of customer in the above protocol. It is important to verify that Alice can bring about payment, else she most likely (unless the merchant is benevolent) will not be able to take advantage of the merchant's commitment.

Notice how the notion of compliance with a protocol helps decouple one agent's specification from another agent's. All Alice wants is a commitment from some merchant that it will deliver the goods. In other words, if an agent commits to another for something, from the perspective of the latter, it does not matter much what the former's goals are or how the former will act to bring about the goal he committed to.

Our primary contribution in this paper is that we formalize the semantic relationship between an agent specification—in terms of goals—and protocols—in terms of commitments. This helps us verify an agent specification with respect to protocols.

This work is motivated by AOSE frameworks such as Tropos [5], where design begins with stakeholder goals and proceeds through a refinement process to identify and characterize alternative designs (plans) that can fulfill these goals. The Tropos framework has been formalized for goals and their refinements [19], but not for goal fulfillment in a multiagent setting where commitments form the primary vehicle for goal fulfillment. We have striven to keep our proposal generic so that it applies not only to Tropos but also other frameworks where there is a need to reason with a collection of agents along with their goals and commitments.

The rest of the paper is organized as follows. Section 2 delineates our conceptual model in terms of agent specifications and protocols: it gives some background, highlights the relationship between them, and introduces the syntax for them. Section 3 motivates the kinds of reasoning we want to perform with examples; it illustrates the subtleties of our approach. Section 4 introduces the formal semantics, and illustrates its applications on some examples introduced earlier. Section 5 concludes the paper with the discussion of related work.

2. CONCEPTUAL MODEL

The principal elements of our framework are agent specifications understood in terms of goals and protocols understood in terms of commitments. In this section, we present some background regarding these concepts, and motivate the need to understand them in a common framework.

2.1 Protocols

A *protocol* is a specification of interaction among agents. A protocol specifies the messages exchanged and how they affect the commitments among the agents [23]. We distinguish protocols from *choreographies*; choreographies specify only the ordering among the messages exchanged without consideration of their meaning in terms of commitments.

A commitment C(Debtor, Creditor, antecedent, consequent) means that the debtor is committed to the creditor for the consequent if the antecedent holds. A commitment is made in a certain sociolegal context and represents a contractual relationship between the debtor and the creditor. A commitment is *discharged* when its consequent is achieved; it is *detached* when the antecedent holds. An unconditional commitment is one where the antecedent is $\top$ (true). A commitment can be *created*, *released*, *canceled*, *delegated*, *assigned*.

A protocol is specified in terms of roles; at runtime, agents would adopt roles in the protocol. For instance, a purchase protocol may specify that the message *Offer* from Merchant to Customer means C(Merchant, Customer, paid, delivered)—the Merchant commits to the Customer that if the payment is made, then the goods will be delivered. When paid holds, the above commitment is detached causing C(Merchant, Customer, $\top$, delivered) to hold. When delivered holds, the Merchant's commitment is discharged.

Instead of specifying protocols in terms of domain specific messages and their meanings in terms of commitments, we assume standard messages pertaining to the commitment operations [7]. For example, *Offer* and its meaning could be captured by the message Create(Merchant, Customer, paid, delivered). Thus, upon sending the create message, the merchant infers C(Merchant, Customer, paid, delivered); upon receiving the message the customer also infers it. We use Declare messages to model the bringing about of propositions such as paid and delivered; for example Declare(Customer, Merchant, paid) is a message from the customer to the merchant. Traditional protocol specifications also include messages for making request for commitments; we model these via *Request* messages. We will use these messages in Section 4 to demonstrate how an agent may reason about communication. We do not consider the other commitment operations in this paper.

Table 1 shows a protocol specification involving four roles: Merchant, Customer, Bank, and Shipper. Messages need not be specified since they are assumed standard (as discussed above). Interestingly, the protocol does not contain any ordering constraints; this reflects new results concerning concurrent updates to commitments [7].

c_{C}: C(Merchant, Customer, paid, delivered)
c_{B}: C(Bank, Merchant, paid, confirmed)
c_{S}: C(Shipper, Merchant, paidShipping, delivered)

Table 1: A protocol depicting a PURCHASE scenario (the labels are for reference purposes)

It is important to dwell on what the protocol specification of Table 1 says. It simply says that the agents that adopt roles in this protocol are *willing* (read *likely*) to engage via the stated commitments. The protocol does not imply that by simply adopting roles in this protocol, the agents will, or even have to, become committed as stated. The commitments themselves would come about at runtime via exchange of messages; whether an agent sends a particular message is solely its own decision. Presumably, protocols specifications will be available from a repository. If an agent wants to play Merchant, and it finds the commitments pertaining to the role appropriate to its needs, it may adopt the role.

We now introduce protocols formally. In the following, x, y, etc. are agents; ψ is a finite set of propositional symbols; α, α_0, β, β_0 etc. are symbols in ψ; $\top$ and $\bot$ are the constants for truth and falsity, respectively; $\wedge$, $\vee$, $\neg$, and $\rightarrow$ are the standard propositional connectives; p, q, etc. are propositions over ψ using the connectives; $\models$ and $\vdash$ denote standard propositional entailment and deduction, respectively.

A commitment is a formula of the form $\mathsf{C}(x, y, p, q)$. Definition 1 expresses a deductive strength relation between commitments [7].

DEFINITION 1. *$\mathsf{C}(x, y, r, u)$ is stronger than $\mathsf{C}(x, y, s, v)$, denoted by $\mathsf{C}(x, y, r, u) \succeq \mathsf{C}(x, y, s, v)$, iff $s \models r$ and $u \models v$.*

Thus, for example (assuming uniform debtors and creditors), $\mathsf{C}(\$10, \mathsf{shoes} \wedge \mathsf{socks}) \succeq \mathsf{C}(\$10, \mathsf{socks})$ (stronger because the debtor promises more in return for the same price); $\mathsf{C}(\$10 \vee \mathsf{coupon}, \mathsf{shoes}) \succeq \mathsf{C}(\$10, \mathsf{shoes})$ (stronger because the debtor offers more ways of obtaining shoes); $\mathsf{C}(\$10, \mathsf{shoes}) \succeq \mathsf{C}(\$10 \wedge \mathsf{coupon}, \mathsf{shoes})$ (stronger because the debtor imposes fewer conditions for obtaining shoes); and $\mathsf{C}(\$10, \mathsf{shoes}) \succeq \mathsf{C}(\$10, \mathsf{shoes} \vee \mathsf{socks})$ (stronger because the debtor offers with more certainty).

Definition 2 distinguishes between a protocol specification and the *effective* protocol induced from it.

DEFINITION 2. *A protocol specification Π is a finite set of commitments. Π^* is Π closed under $\succeq$; we refer to Π^* as the effective protocol specification.*

The effective protocol represents all the commitments that may actually come about, and is in effect the protocol. We consider the effective protocol based on the reasonable assumption that if a debtor were willing to make some commitment, then it would have no problem making a weaker commitment. For example, assume a merchant is willing to make the commitment $\mathsf{C}(\$10, \mathsf{shoes} \wedge \mathsf{socks})$; it would be reasonable to assume that he'd also be willing to make just $\mathsf{C}(\$10, \mathsf{shoes})$. The consideration of the closure under $\succeq$ may be viewed as a kind of normalization for protocol specifications: the sets $\{\mathsf{C}(\$10, \mathsf{shoes} \wedge \mathsf{socks})\}$ and $\{\mathsf{C}(\$10, \mathsf{shoes} \wedge \mathsf{socks}), \mathsf{C}(\$10, \mathsf{shoes})\}$ yield the same protocol.

2.2 Goals

Goals represent the motivational component of agent reasoning, and have been extensively applied in both multiagent systems and requirements engineering. For the purposes of this paper, we apply goals as have been applied extensively in AOSE, especially in Tropos [5], an influential AOSE methodology.

Following Tropos, we represent stakeholders as agents, and the stakeholders' goals are identified with the respective agents' goals. The goals of the agents are analyzed and modeled using goal *decomposition*, which involves understanding goals in a structural manner, and contributions, which involves understanding the impact of goals on each other. Decomposition is of two types. An AND decomposition of a goal into constituents means that all constituent goals have to be achieved to achieve the goal. An OR decomposition means that at least one constituent must be achieved to achieve the goal. Following [19], contributions between goals in Tropos are of the following four types. $++S(g, g')$ means that achieving g achieves g'; $--S(g, g')$ means that achieving g denies the achievement of g'; $++D(g, g')$ means denying the achievement of g denies the achievement g'; $--D(g, g')$ means that denying the achievement of g achieves g'. An agent may have the *capability* to achieve some goals itself; for others, it has to depend on others.

Below, we formally introduce the goal models our framework supports.

Syn$_1$. $G \xrightarrow{AND} \alpha$, where $G \subseteq \psi$ and G is not empty.
It means that achieving all the goals represented by each of the symbols in G is a way of achieving α.

Syn$_2$. $G \xrightarrow{OR} \alpha$, where $G \subseteq \psi$ and G is not empty.
It means that achieving a goal represented by one of the symbols in G is a way of achieving α.

Syn$_3$. $\alpha \rightarrow \beta$, equivalent to $++S(\alpha, \beta)$.

Syn$_4$. $\alpha \rightarrow \neg\beta$, equivalent to $--S(\alpha, \beta)$

Syn$_5$. $\neg\alpha \rightarrow \beta$, equivalent to $--D(\alpha, \beta)$

Syn$_6$. $\neg\alpha \rightarrow \neg\beta$, equivalent to $++D(\alpha, \beta)$

An agent's goal model consists of zero or more statements, each conforming to one of the schemas **Syn$_1$**–**Syn$_6$**. In addition, all the symbols in ψ are implicitly in the goal model (possibly undecomposed). The idea is that ψ represents all the goals that an agent is interested in; queries about particular goals represent those to be considered for achievement.

We impose the syntactic restriction on goal models that there are no cyclic decompositions; such a goal model would not make any sense. Thus, for example, if α is decomposed (AND or OR) into α_0 (among other goals), then α_0 cannot be decomposed into α (among other goals).

An agent's capability set $\mathbf{C}$ is a subset of ψ: the agent has the capability for some goal α if and only if α belongs to the set.

Table 2 shows an agent Rob's goal model. Rob's goal is to sell books (sold). The sell books goal is AND-decomposed into goals receiving payment (receivedPayment) and shipping (shipped) the books. The goal receivedPayment is AND-decomposed into getting the payment (paid) and getting a confirmation for the payment (confirmed). shipped is AND-decomposed into paying for the shipping (paidShipping) and delivering the books (delivered). Rob has the capability to pay for the shipping, that is, to achieve paidShipping.

a_0: $\{\mathsf{receivedPayment}, \mathsf{shipped}\} \xrightarrow{AND} \mathsf{sold}$;
a_1: $\{\mathsf{paid}, \mathsf{confirmed}\} \xrightarrow{AND} \mathsf{receivedPayment}$;
a_2: $\{\mathsf{paidShipping}, \mathsf{delivered}\} \xrightarrow{AND} \mathsf{shipped}$;
$\mathbf{C} = \{\mathsf{paidShipping}\}$

Table 2: Rob's goal model specification; the prefixes a_i are for reference purposes

It is important to understand the difference between decomposition and contribution. Decomposition takes into account intentionality on part of the agent; contribution does not. Let's say $\{\alpha, \beta\} \xrightarrow{AND} \gamma$. Achieving α and β does not achieve γ—unless one achieved them both intentionally in order to achieve γ. However, $\alpha \wedge \beta \rightarrow \gamma$ disposes with intentionality; it simply says that if α and β are achieved, it *means* that γ is achieved too. Moreover, one cannot take into account contributions from those goals that are not already part of some decomposition. In other words, contributions are treated as side-effects. Thus, for example, if an agent's goal is α, and $\beta \rightarrow \alpha$, then achieving β does not constitute a way to achieve α, unless β is also a goal that the agent wants to achieve anyway. Example 1 illustrates this.

EXAMPLE 1. *To bake cookies, an agent will intentionally turn on the oven; as a side-effect the temperature in the room*

will increase. However, to increase the room temperature, the agent won't turn on the oven.

3. CONNECTING AGENT SPECIFICATIONS WITH PROTOCOLS

An agent's goal model is a specification $\langle \mathbf{G}, \mathbf{C} \rangle$ such that $\mathbf{G}$ is a set of statements of the form $\mathbf{Syn_1}$–$\mathbf{Syn_6}$, and $\mathbf{C}$ is the set of capabilities.

We formalize the notion of an agent adopting a role in a protocol. Let ρ be a role in a protocol Π that x wants to adopt. $\Pi.\rho(x)$ is the protocol Π except that x is bound to role ρ in Π: wherever ρ occurs in the protocol, it is replaced by x. $\Pi^*.\rho(x)$ is the corresponding effective protocol. We refer to the triple $\langle \langle \mathbf{G}, \mathbf{C} \rangle_x, \Pi.\rho(x) \rangle$ as x's *protocol-bound* specification.

If an agent wants to achieve a certain goal, but does not have the capability for it, then the only way to possibly achieve it is by interacting with others. This leads to the formulation of the problem this paper addresses: *given an agent's protocol-bound specification and a goal query, we want to verify if the protocol supports achieving the goals represented by the query.* This verification is ideally done *a priori*, that is, before enacting the protocol. This verification may be done by the agent designer or at runtime by the agent itself.

Consider Rob as specified in Table 2; let's say we want to verify that Rob playing Merchant in PURCHASE (as specified in Table 1) supports his goal sold, in other words, whether $\langle \langle \mathbf{G}, \mathbf{C} \rangle_{Rob}, \Pi.\mathsf{Merchant}(Rob) \rangle$ supports sold. The goal sold is decomposed into leaf-level goals: paid, confirmed, delivered, paidShipping. Rob has no capability to bring about paid, confirmed, delivered; he has capability to bring about paidShipping. Rob can make a commitment to some customer for delivered if the customer pays, thus supporting paid with c_{C}. He *hopes* to get a commitment from the bank upon request—he can no more than hope as the bank is autonomous—that it will send him confirmation, thus bringing about confirmed if he notifies the bank of payment. In this manner, he supports confirmed via c_{B}. He hopes to get a commitment from the shipper that books will be delivered if he pays for the shipment (paidShipping, for which he has the capability), thus supporting delivered via c_{S}. Thus, all leaf-level goals are supported, which means that $\langle \langle \mathbf{G}, \mathbf{C} \rangle_{Rob}, \Pi.\mathsf{Merchant}(Rob) \rangle$ supports sold.

The above is illustrative of the reasoning we wish to formalize, but it is a simple example. In general, an agent's protocol-bound specification may not support the achievement of certain goals. This may be because the protocol is too weak to support the agent's goal or because of conflicts arising from the consideration of certain goals.

Below we introduce some intuitions about reasoning with protocol-bound specifications with the help of examples. In Section 4 we will formalize these intuitions. We distinguish between two primary modes of reasoning: exploiting commitments for goal support and exploiting goals for commitment support. The latter relates to compliance with protocols.

3.1 Exploiting commitments for goal support

Given an agent's protocol-bound specification and a set of goals from the model that are being considered for achievement, we would want to make sure that the protocol supports achieving those goals.

Commitments can support goals in two primary ways. Consider an agent x. To achieve a goal g, x can play the debtor in a commitment: x commits to another agent y that if y brings about g, then x will achieve some other goal g' for y. In other words, $\mathsf{C}(x, y, g, g')$ supports x's goal g. We refer to this way of exploiting commitments *antecedent support*. Example 2 illustrates this notion.

EXAMPLE 2. *c_{C} supports Rob's goal* paid.

Alternatively, to achieve g, x could play the creditor in a commitment: x hopes that some other agent y would commit to x that if x brings about some goal g', then y will bring about g for x. However, there is a slight caveat here: y is unlikely to bring about g unless x achieves g'. In other words, $\mathsf{C}(y, x, g', g)$ supports x's goal g only if x can achieve g'. We refer to this way of exploiting commitment *consequent support*. Example 3 illustrates this notion. In the examples below, we assume that agents are bound to Merchant in PURCHASE unless otherwise specified.

EXAMPLE 3. *c_{S} supports Rob's goal* delivered *because Rob has the capability to bring about the antecedent* paidShipping.

Example 4 is a special case of consequent support where the antecedent itself is supported by a commitment from another agent. An agent can take advantage of such commitments when it lacks the capability for the antecedent. The example highlights the *open* nature of multiagent systems.

EXAMPLE 4. *c_{B} supports Rob's goal* confirmed. *Although Rob does not have the capability for* paid, paid *is antecedent-supported by c_{C}.*

If a commitment's antecedent is stronger than an agent can support, then that commitment cannot support a goal. Example 5 shows this.

EXAMPLE 5. *Suppose c_{S} is replaced by $c_{\mathsf{S\_INS}} = \mathsf{C}(\mathsf{Shipper}, \mathsf{Merchant}, \mathsf{paidShipping} \wedge \mathsf{paidInsurance}, \mathsf{delivered})$ in the specification of* PURCHASE *in Table 1. Then, Rob cannot exploit $c_{\mathsf{S\_INS}}$ to support* sold*: he does not have the capability for* paidInsurance.

In general, we are interested in knowing if an agent can achieve certain goals together. Example 6 shows how querying for conjunctive goals may lead to a conflict.

EXAMPLE 6. *Suppose Rob's goal model specification is augmented with a goal* taxEvaded *along with the contribution* confirmed $\rightarrow \neg$taxEvaded. *In other words, getting confirmation of transactions from a bank implies that Rob cannot evade tax (presumably because of accounting regulations imposed by the government). Therefore, the conjunctive goal* sold $\wedge$ taxEvaded *is not supported.*

There is a caveat in antecedent support: achieving the antecedent might deny some other goal. Example 7 highlights this.

EXAMPLE 7. *Consider that in* PURCHASE*, c_{C} is replaced by $c_{\mathsf{C\_REG}} = \mathsf{C}(\mathsf{Merchant}, \mathsf{Customer}, \mathsf{paid} \wedge \mathsf{registered}, \mathsf{delivered})$. Also suppose Rob has two additional goals* registered *and*

bookkeepingEliminated, *and that that achieving* registered *denies* bookkeepingEliminated. *Suppose, we query whether* sold ∧ bookkeepingEliminated. *The answer is negative—to support* delivered, registered *needs to be supported (antecedent support as discussed above); however* registered *denies goal* bookkeepingEliminated.

a_0; a_1; a_2; a_3: {testDone, projectDone} $\xrightarrow{OR}$ passed;
a_4; projectDone → ¬delivered;
$\mathbf{C}$ = {projectDone, paidShipping}

Table 3: Sam's goal model and capability specification. a_0–a_2 are from Table 2

There might be more than one way to achieve a particular goal. Example 8 shows how alternatives may be exploited to support goals.

EXAMPLE 8. *Consider an agent Sam, whose goal model is similar to Rob's, except for an additional goal exam* passed. *Table 3 shows Sam's goal model specification. The achievement of goal* projectDone *is time consuming and leaves him little time to deal with shippers; as such it contributes negatively to the achievement of* delivered *(a_4). Sam has the capability for* projectDone, *but not for* testDone. *Suppose we want to verify whether playing* Merchant *in* PURCHASE *supports Sam's conjunctive goal:* sold ∧ passed. *The answer is no—*projectDone *denies* delivered. *However, if Sam also had the capability for* testDone, *instead of doing the project, Sam could choose to take the test to pass the exam; then both* sold *and* passed *would be supported.*

3.2 Exploiting goals for commitment support

This relates to compliance with commitments; that is, the fulfillment of commitments. For any commitment that an agent could possibly make, it makes sense to verify that the agent supports the fulfillment of the commitment. This means checking for commitment support reduces to checking for support of the consequent of the commitment. Example 9 illustrates this notion.

EXAMPLE 9. *Consider Rob from Table 2 playing* Merchant *in* PURCHASE. *Rob's designer would want to be sure that Rob's commitment c_C is supported—that he would be able to bring about* delivered. *In Rob's case,* delivered *is supported by another commitment c_S—he can achieve* paidShipping *and hopes that the shipper will bring about the consequent.*

It is important to note that the reasoning highlighted in Example 9 does not guarantee compliance—that at runtime, Rob will fulfill such a commitment. Many things could go wrong at runtime: Rob could fail or act in error, the communication infrastructure might fail, or, more pertinently, Rob might be relying on the fulfillment of some other commitment (such as from the supplier of the *item*) and that commitment might go unfulfilled. Compliance with commitments, in any reasonably complex multiagent system, can only be determined at runtime. What we want to accomplish here is to verify that an agent is correctly designed modulo all that is not under its control.

Example 10 shows an example where a commitment is not supported.

EXAMPLE 10. *Consider Sam from Table 3 playing role* Merchant *in* PURCHASE. *Consider the fulfillment of c_C; Sam can fulfill it by bringing about* delivered. *Let's say he wants to pass the exam as well (*passed*). To pass the exam, Sam can achieve* projectDone*; however that negatively contributes to* delivered. *Thus, in this example fulfillment of c_C is not supported.*

4. SEMANTICS

We now describe a semantics that ties the reasoning about goals with commitments; it generalizes the intuitions presented in Section 3. Throughout this section, we use the term "specification" to refer to an agent's protocol-bound specification. Definition 3 formalizes the notion of an interpretation.

DEFINITION 3. *An interpretation $\mathcal{I}$ is an assignment of truth values ($\top$ or $\bot$) to all symbols in ψ.*

Let $\psi = \{\alpha, \beta\}$; there are four possible interpretations corresponding to the different truth assignments to α and β. We can equivalently write the interpretation as formulas; for example, $\mathcal{I}_0 = \{\alpha = \top, \beta = \top\}$ is the formula $\alpha \wedge \beta$, and $\mathcal{I}_1 = \{\alpha = \top, \beta = \bot\}$ is $\alpha \wedge \neg\beta$.

$\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} p$ signifies that the protocol-bound specification of x supports achieving p under the interpretation $\mathcal{I}$. (Below, we use y as a variable over roles.) $\mathbf{G}_{\mathcal{C}}$ denotes the statements in $\mathbf{G}$ conforming to Schemas $\mathbf{Syn_3}$–$\mathbf{Syn_6}$ (the contributions). The relation $|\models^{\mathcal{I}}$ is computed according to the rules $\mathbf{Sem_1}$–$\mathbf{Sem_5}$.

$\mathbf{Sem_1}$. $\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} \top$

$\mathbf{Sem_2}$. $\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} \alpha$ if and only if $\mathbf{G}_{\mathcal{C}} \cup \mathcal{I} \not\models \neg\alpha$, and:

(i.) $\alpha \in \mathbf{C}$, or

(ii.) $G \xrightarrow{AND} \alpha \in \mathbf{G}$

$$\forall g \in G : \ \langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} g, \text{ or}$$

(iii.) $G \xrightarrow{OR} \alpha \in \mathbf{G}$

$$\exists g \in G : \ \langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} g, \text{ or}$$

(iv.) $\exists \mathsf{C}(y, x, s, \alpha) \in \Pi^*.\rho(x)$ such that

$$\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} s, \text{ or}$$

(v.) $\exists \mathsf{C}(x, y, s, u) \in \Pi^*.\rho(x)$ such that $s \models \alpha$

$\mathbf{Sem_3}$. $\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} q \vee r$ if and only if

(i.) $\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} q$ or $\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} r$, or

(ii.) $\exists \mathsf{C}(y, x, s, q \vee r) \in \Pi^*.\rho(x)$ such that

$$\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} s \text{ and } \mathbf{G}_{\mathcal{C}} \cup \mathcal{I} \not\models \neg(q \vee r)$$

$\mathbf{Sem_4}$. $\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} q \wedge r$ if and only if

$$\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} q \text{ and } \langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} r$$

$\mathbf{Sem_5}$. $\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} \neg q$ if and only if

$$\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \not\models^{\mathcal{I}} q$$

Sem$_1$ says that all specifications model $\top$ under any interpretation.

Sem$_2$ says that an agent's specification supports a symbol under an interpretation if and only if the constraints do not derive a contradiction (considering the contributions), and one of the following conditions holds (with numbers corresponding to those in **Sem$_2$**).

(i.) The symbol belongs to the capabilities, meaning that the agent can bring it about itself.

(ii.) There is an AND-decomposition of the symbol such that the specification supports each constituent of the decomposition under the interpretation.

(iii.) There is an OR-decomposition of the symbol such that the specification supports at least one constituent of the decomposition under the interpretation.

(iv.) There exists a commitment *to* the agent for the atomic proposition, and the specification supports its antecedent under the interpretation.

(v.) There exists a commitment *from* the agent to some other that if the other agent brings about a proposition that entails the atomic proposition.

Sem$_3$ handles the case of a disjunctive goal. It describes two ways of supporting such a goal.

(i.) One of the disjuncts is supported.

(ii.) There is a commitment to the agent for the disjunctive goal such that the antecedent can be supported, and $\mathbf{G}_{\mathcal{C}} \cup \mathcal{I}$ does not derive the negation of the disjunction.

Sem$_4$ handles the case of a conjunctive goal. A conjunctive goal is supported if and only if the conjuncts are supported.

Sem$_5$ says that a negated goal proposition is supported if and only if the nonnegated goal proposition is not supported.

Let's pay closer attention to clauses **Sem$_2$**(iv.) and (v.) with respect to compliance. **Sem$_2$**(iv.) refers to a commitment from another role y; so the question of x being compliant with the commitment does not arise. Agent x could worry about y's compliance; however, y's construction is opaque to x (as would be in any open system). **Sem$_2$**(v.) refers to a commitment from x; however, even there we do not check whether the specification supports x's compliance with the commitment. The reason we don't do it in (v.) is to enable modular reasoning. While some agents would be *prudent* enough to check for compliance support, there may be *reckless* agents who are interested in satisfying their own goals, but not their commitments to others. Hence, we define commitment support separately in Definition 4.

DEFINITION 4. *An agent x with protocol-bound specification $\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle$ satisfies commitment $\mathsf{C}(x, y, r, u)$ if and only if $\langle\langle \mathbf{G}, \mathbf{C}\rangle_x, \Pi.\rho(x)\rangle| \models^{\mathcal{I}} u$*

Thus, given an interpretation, a designer may check for the support of a commitment by simply querying for the consequent of the commitment. In the query, he may also include other goals if needed (including those considered for the fulfillment of other commitments).

4.1 Applying the semantics

Let's see now how the semantics applies to some queries concerning agent Sam as specified in Table 3.

QUERY 1. *Can Sam support goal* sold *while playing role* Merchant*?*

Let's consider $\mathcal{I} = \{$sold, receivedPayment, shipped, paid, confirmed, paidShipping, delivered$\}$ (only symbols with valuation $\top$ are shown in the interpretation). We ask whether $\langle\langle \mathbf{G}, \mathbf{C}\rangle_{Sam}, \Pi.\rho(Sam)\rangle| \models^{\mathcal{I}}$ sold. Goal sold is a symbol, so **Sem$_2$** applies. We check whether $\mathbf{G}_{\mathcal{C}} \cup \mathcal{I}$ contradicts sold—it does not. Clause (i.) cannot be used as Sam is not capable of sold; (iii.) does not apply since the goal is not OR decomposed; neither (iv.) or (v.) apply as there is no suitable commitment in $\Pi^*.\rho(Sam)$. Clause (ii.) applies though; so we recursively check for each of the goals receivedPayment and shipped. Let's consider receivedPayment first. Again, **Sem$_2$** applies. receivedPayment is not contradicted, and (ii.) applies. So we recursively check for each of the goals paid and confirmed. Again, **Sem$_2$** applies to each; neither is contradicted. By **Sem$_2$**(v.), paid is supported by Sam's commitment c_{C}. By **Sem$_2$** (iv.), we get that confirmed is supported by c_{B}; however we must recursively check to see if the antecedent paid of c_{B} is supported, which already is. Thus, receivedPayment is supported. Let's consider shipped now. Again, **Sem$_2$** applies: shipped is not contradicted, and (ii.) applies. So we recursively check for each of the goals paidShipping and delivered. Again, **Sem$_2$** applies to each; neither is contradicted. By **Sem$_2$**(i.), paidShipping is supported, since Sam has the capability for it. By **Sem$_2$** (iv.), we get that delivered is supported by c_{S}; however, we must recursively check to see if the antecedent paidShipping is supported, which already is. Thus, shipped is supported. And thus, sold is supported.

QUERY 2. *Can Sam support goals* sold$\wedge$passed *while playing role* Merchant*?*

Let's try with $\mathcal{I}' = \{$sold, receivedPayment, shipped, paid, confirmed, paidShipping, delivered, passed, projectDone$\}$. Answering this query amounts to the following: $\langle\langle \mathbf{G}, \mathbf{C}\rangle_{Sam}, \Pi.\rho(Sam)\rangle| \models^{\mathcal{I}'}$ sold $\wedge$ passed. The query is a goal conjunction; therefore, **Sem$_4$** applies and we recursively check each conjunct (sold and passed). Goal sold is reasoned about as shown above, except that when applying **Sem$_2$** to delivered, we derive a contradiction. Indeed, goal projectDone is true in $\mathcal{I}'$ and statement *a4* in Table 3 is the conflict source. Hence, sold $\wedge$ passed are not supported.

Let's try with another interpretation where projectDone is false: $\mathcal{I}'' = \{$sold, receivedPayment, shipped, paid, confirmed, paidShipping, delivered, passed, testDone$\}$. **Sem$_4$** applies and tells to check $\mathcal{I}''$ for sold and passed. Let's start with passed. **Sem$_2$** (iii.) applies, since passed is OR-decomposed to testDone and projectDone. The constraints do not contradict passed, thus we check whether $\mathcal{I}''$ supports at least one subgoal. Let's try testDone; it is not denied by $\mathbf{G}_{\mathcal{C}} \cup \mathcal{I}''$, but no sub-clause of **Sem$_2$** is applicable. Thus, let's test projectDone. Unfortunately, projectDone is contradicted—it is not in $\mathcal{I}''$. Therefore, $\mathcal{I}''$ does not support the queried goal proposition.

In order to answer the query, we should try all possible interpretations. None of those would support the query, meaning that Sam cannot support the goal query.

4.2 Enactment

So far, we have talked about goals being supported; let's see how an agent would actually communicate to achieve its goals. Figure 1 explicitly shows a possible enactment for Rob—as specified in Table 2—playing Merchant in PURCHASE using an interaction diagram, and is annotated with commitments at the points agents send and receive messages. c_{UB}, c_{UC}, and c_{US} are the unconditional commitments corresponding to c_B, c_C,c_S, respectively. Arrows with empty heads represent those messages that Rob has no control over; those with filled heads represent messages Rob can send anytime. Thus, for example, Declare(paidShipping) has a filled arrowhead; however, Declare(paid), Declare(delivered), Declare(confirmed) have empty arrowheads. The figure highlights the fact that *a priori* we can only verify whether Rob's goals are supported; if the agents playing the other roles act in a way conducive to Rob's goals, only then Rob can achieve his goals. For example, Rob can request the bank for c_B; however, there is no guarantee that the bank will create such a commitment. Similarly, Rob can pay the shipper, however there is no guarantee that the shipper will deliver. However, in the enactment of Figure 1, everything goes in a manner conducive to Rob's goal sold.

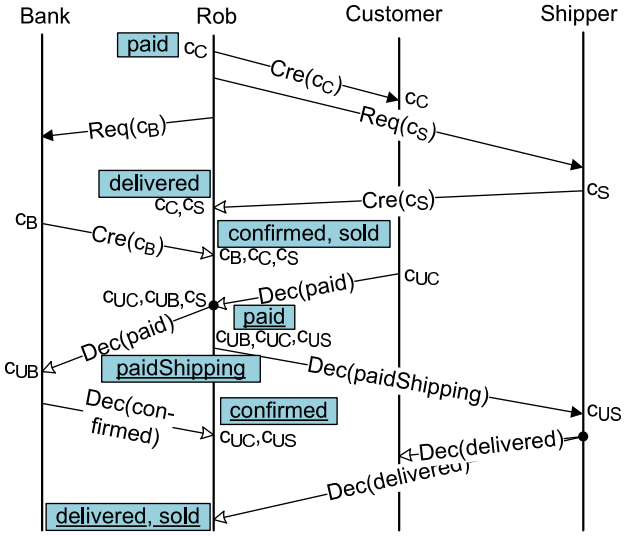

Figure 1: An enactment that achieves Rob's goals using Merchant in PURCHASE. Cre is abbreviation for Create; Req for Request; Dec for Declare. Rob's lifeline is annotated with its goals as they are supported (without underlining) and achieved (with underlining)

Rob acts like this to achieve sold: to achieve paid, he offers c_C to customer (by **Sem₂**(v.)); to achieve confirmed, he requests c_B from the bank (by **Sem₂**(iv.)), and to achieve delivered he requests c_S from the shipper (again, by **Sem₂**(iv.)). Upon getting Declare(paid) from the customer, Rob forwards it to the bank for confirmation (by the rule that creditors must notify debtors of the detach of a commitment [7], here c_B). Rob sends Declare(paidShipping) to the shipper; the shipper presumably sends Declare(delivered) to the customer and forwards the message to Rob (by the rule that debtors must notify creditors of the discharge of a commitment [7], here c_S). Presumably, the bank eventually sends Declare(confirmed). Thus all of Rob's goals are achieved.

5. DISCUSSION

In the preceding, we encoded agents and protocols in terms of goals and commitments respectively, and demonstrated and formalized the semantic relationship between them. The formalization is based on the insight that in open multiagent systems agents would need to either make commitments to or get them from others in order to meet their goals. We introduced the notion of a protocol-bound agent specification to express the notion of an agent adopting a role in a protocol, and used the relationship to verify if an agent's goals and commitments are supported by the specification. In general, agents and protocols would be *independently specified*: agents would be specified by their stakeholders, and protocols, being reusable, may be available from repositories. A deployed agent could reason about its own protocol-bound specification in order to select the *right* protocol.

In reasoning about protocol-bound specifications, we tied together the *cognitive* with the *social*, the *private* with the *public*, and *agent reasoning* with *communication* [20, 18, 12], long-standing concerns of multiagent systems research. More importantly, given that goals express requirements [5] and protocols express architectural connections [7], we have also formally tied requirements with architecture. We conclude with a discussion of the literature and some possible directions for further research.

Singh [20] showed in unequivocal terms that communication between agents in open multiagent systems cannot be modeled in terms of cognitive primitives, including goals. An agent may be designed in any programming language—in Java, BDI-based, rule-based, and so on; however, its communications would be judged correct based on the fulfillment of its commitments.

Most approaches for agent reasoning do not consider protocols as independent artifacts. Dignum *et al.* [13] consider how norms and obligations imposed upon an agent impact agent reasoning. They formalize these concepts in a modal logic and consider preference orderings over states to reason about action. In their approach, an agent acts according to the indicated preferences, possibly choosing to violate obligations and ignoring norms. An agent may choose to do that; nonetheless, it is important to understand where problems could occur—which goals might go unsatisfied and which commitments may be violated. As an analogy, consider that just because an agent may willfully violate a commitment does not obviate the value of compliance as a correctness criterion for interactions. Alechina *et al.* [1] specify agents in terms of beliefs, goals, and plans. They analyze agent specifications for correctness in terms of liveness and correctness with respect to deliberation strategies.

Dastani *et al.* [10] consider protocols; they check agents against roles to verify whether agents can enact the roles. As such, their motivations and the spirit of their approach are similar to ours. However, they specify roles in terms of goals, not in terms of commitments, making their approach less suitable for open multiagent systems.

Prominent AOSE methodologies model interaction via messaging (often as a variant of UML), without considering high-level abstractions. Tropos [5] supports the derivation of UML interaction diagrams from requirements expressed as goals. Gaia [24] emphasizes organization metaphors such as responsibilities; however those are encoded as procedures.

Cheong and Winikoff [6] derive interaction protocols in terms of action sequences (that are mapped to communication in the form of messages) from requirements specified as goals. The goals are at the level of a multiagent system, not any particular agent's goals. By contrast, in our approach, the goals are of a particular agent, and we verify if the role that the agent wants to play in some chosen protocol is compatible with the agent's goals.

Existing work on protocols and commitments has largely focused on problems of specification and verification, not on role adoption by individual agents [15, 4, 2, 3]. Artikis *et al.* [2] do consider constraints that an agent must satisfy to adopt a role; however those are not the same as an agent's goals. Winikoff [22] implements reasoning about commitments (such as for detach and discharges), in a BDI-based agent programming language, and formalized the update relations between beliefs and commitments via inference rules.

Endriss *et al.* [14] specify protocols and agents declaratively in terms of if-then rules, and consider compliance with such protocols. They do not use any agent-oriented abstraction in encoding the protocols and agents.

The value of considering commitments in conjunction with goal models has been noted in [17]. Telang and Singh [21] enhance Tropos with a notation and methodological elements for constructing business models rooted in goals and commitments. Our work expands on the same basic theme by formalizing some of the notions involved.

Promising directions of future work include devising strategy-computing algorithms for agents (for example, reckless vs. prudent as mentioned before); treating commitment-related predicates themselves as goals (such as violated, for example) and to express preferences over them as in [13]; supporting monitoring of goals and commitments as in [9] so that agents may revise strategies at runtime in response to failures or under-performance.

Acknowledgments. This work was partially supported by FP6-EU project SERENITY contract 27587.

6. REFERENCES

[1] N. Alechina, M. Dastani, B. S. Logan, and J.-J.Ch.Meyer. Reasoning about agent deliberation. In *Proceedings, KR*, pages 16–26, 2008.

[2] A. Artikis, M. J. Sergot, and J. V. Pitt. Specifying norm-governed computational societies. *ACM Transactions on Computatinal Logic*, 10(1), 2009.

[3] M. Baldoni, C. Baroglio, A. K. Chopra, N. Desai, V. Patti, and M. P. Singh. Choice, interoperability, and conformance in interaction protocols and service choreographies. In *Proceedings, AAMAS*, pages 843–850, 2009.

[4] J. Bentahar, J.-J. Meyer, and W. Wan. Model checking communicative agent-based systems. *Knowledge-Based Systems*, 22(3):142–159, 2009.

[5] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multiagent Systems*, 8(3):203–236, 2004.

[6] C. Cheong and M. P. Winikoff. Hermes: Designing flexible and robust agent interactions. In *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages 105–139, 2009.

[7] A. K. Chopra and M. P. Singh. Multiagent commitment alignment. In *Proceedings, AAMAS*, pages 937–944, 2009.

[8] R. Conte, R. Falcone, and G. Sartor. Introduction: Agents and norms: How to fill the gap? *AI and Law*, 7(1):1–15, 1999.

[9] F. Dalpiaz, P. Giorgini, and J. Mylopoulos. An architecture for requirements-driven self-reconfiguration. In *Proceedings, CAiSE*, Volume 5565 of *LNCS*, pages 246–260. Springer, 2009.

[10] M. Dastani, V. Dignum, and F. Dignum. Role-assignment in open agent societies. In *Proceedings, AAMAS*, pages 489–496, 2003.

[11] M. Dastani, J. Hulstijn, F. Dignum, and J.-J. C. Meyer. Issues in multiagent system development. In *Proceedings, AAMAS*, pages 922–929, 2004.

[12] F. Dignum. Autonomous agents with norms. *AI and Law*, 7(1):69–79, 1999.

[13] F. Dignum, D. Morley, E. Sonenberg, and L. Cavedon. Towards socially sophisticated BDI agents. In *Proceedings, ICMAS*, pages 111–118, 2000.

[14] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Protocol conformance for logic-based agents. In *Proceedings, IJCAI*, pages 679–684, 2003.

[15] N. Fornara, F. Viganò, M. Verdicchio, and M. Colombetti. Artificial institutions: A model of institutional reality for open multiagent systems. *AI and Law*, 16(1):89–105, 2008.

[16] K. V. Hindriks and M. B. van Riemsdijk. A Computational Semantics for Communicating Rational Agents Based on Mental Models. In *Proceedings, ProMAS Workshop*, 2009.

[17] A. U. Mallya and M. P. Singh. Incorporating commitment protocols into Tropos. In *Proceedings, AOSE 2005 Workshop*, volume 3950 of *LNCS*, pages 69–80. Springer, 2006.

[18] J. Pitt and A. Mamdani. A protocol-based semantics for an agent communication language. In *Proceedings, IJCAI*, pages 486–491, 1999.

[19] R. Sebastiani, P. Giorgini, and J. Mylopoulos. Simple and minimum-cost satisfiability for goal models. In *Proceedings, CAiSE*, Volume 3084 of *LNCS*, pages 20–35. Springer, 2004.

[20] M. P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, 1998.

[21] P. R. Telang and M. P. Singh. Enhancing Tropos with commitments: A business metamodel and methodology. In *Conceptual Modeling: Foundations and Applications*, Volume 5600 of *LNCS*, pages 417–435, Springer, 2009.

[22] M. Winikoff. Implementing commitment-based interactions. In *Proceedings, AAMAS*, pages 1–8, 2007.

[23] P. Yolum and M. P. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings, AAMAS*, pages 527–534, 2002.

[24] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering Methodology*, 12(3):317–370, 2003.