7

Interaction Protocols

Matteo Baldoni,Cristina Baroglio, Amit K. Chopra, Akin Günay

ABSTRACT. Multiagent systems are conceptually decentralized: each agent is a locus of control and agents interact with each other via arms-length communication. This makes the study and design of *interaction protocols* a central concern in multiagent systems research. This chapter introduces the idea of an interaction protocol along with a discussion of the two kinds of protocols that have been been most studied, messaging protocols and meaning-based protocols. Whereas messaging protocols capture oeprational constraints on messaging, meaning-based protocols specify what the messages mean in terms of the social state. We discuss AUML and BSPL as alternative ways of specifying messaging protocols and dwell at length on commitment protocols as an exemplar of meaning-based protocols. We also discuss the properties of these protocols and their verification.

1 Introduction

A highly promising application of multiagent systems (MAS) is in domains that involve interaction between autonomous social *principals*—typically, humans and organizations. Business and finance, healthcare, scientific collaboration, and social media are all examples of such domains. Such systems are properly *sociotechnical*, reflecting the use of technology by social principals to carry out social processes [Chopra and Singh, 2016b]. Sociotechnical systems are naturally *decentralized*: each principal is a locus of autonomy and, therefore, decision-making. A central challenge facing the software engineering of sociotechnical systems is accommodating the autonomy of principals in interacting flexibly with each other while at the same time also supporting notions of what it means to interact correctly.

Multiagent systems (MAS) are ideally suited for modeling sociotechnical systems. In the basic model of a MAS, each agent represents an autonomous social principal. The agents interact with each other (on behalf of their principals) to carry out social processes. However, not all interactions are legal. Interaction protocols specify the *rules of encounter* between agents [Singh, 1998]. If an agent follow the rules, it is *compliant* with the protocol; otherwise it is non-compliant. Thus, a protocol specification serves as a standard of correctness for agent behavior in a multiagent system. Protocols also promote autonomy and flexibility because they leave principals free to design the agents as they wish, in accordance with their goals, potentially in a manner that will ensure protocol compliance but not necessarily.

Two kinds of protocols are relevant to MAS:

Messaging protocols describe constraints on message flow between agents, typically in terms of ordering and occurrence constraints. For example, a protocol for scientific collaboration may specify that a request for a resource (message) must precede any response. Such protocols are violated when a messaging constraint is violated. Agent UML [Huget, 2004], BSPL [Singh, 2011a], HAPN [Winikoff *et al.*, 2018] are all proposals along this line. In certain contexts, compliance with such protocols may be enforced that allows only legal messages to be sent, e.g. [Singh, 2011b]. In other cases, deviations will be automatically identified. Messaging protocols are being formalized in many business domains and led to the development of industry-supported standards. These efforts include Intel-led *RosettaNet* (e-business) [RosettaNet, 1998], ABN-AMRO led *TWIST* (foreign exchange transactions) [TWIST, 2006], and *HL7* (healthcare) [HL7, 2002], among many others. The efforts vary in their level of sophistication: some specify only message formats, others specify request-response protocols, whereas others specify only some possible *sequences* of interaction.

Meaning-based protocols describe the social *meanings* of messages in terms of normative expectations such as commitments, prohibitions, and so on. For example, the meaning of the request may be expressed as a commitment by the requester that if the request is granted, then the resource will be released by a specified deadline. In such protocols, noncompliance results from violation of commitments. For example, if the request is granted, but the resource is not released by the deadline. We will consider *commitment protocols* as exemplar of this style of protocols. Compliance of an agent with norms cannot, in general, be guaranteed: it would depend on the design of the agent.

A protocol is an abstraction for interaction. The key benefit is that it captures application-level logic pertaining to interactions in a reusable manner. In principle, protocols may be refined and composed. Naturally, we may want to verify that a protocol has desirable properties. A protocol may be statically verified for properties such as deadlock freedom, fairness, and safety before the interaction starts [El-Menshawy *et al.*, 2011]. Agents may be verified for behavior in accordance with the specification (*conformance*) [Baldoni *et al.*, 2006]. At runtime, the compliance of agents with protocols can be monitored [Chesani *et al.*, 2013]. If the interacting agents conform to the specification of the role they play, the interaction inherits the properties verified on the protocol. For instance, if in a protocol specification roles are proved interoperable, any agent that will conform to the protocol is guaranteed that its interaction with any other agents, playing the other roles and compliant with their specification, will succeed [Rajamani and Rehof, 2002; Bravetti and Zavattaro, 2009].

In this chapter, we give an overview of messaging and commitment protocols with the help of examples specified in important languages. We highlight some of the key features and subtleties of these protocols and discuss the relevant interesting properties and their verification. We also highlight important open challenges in interaction protocol research.

The rest of the chapter is organized as follows. Section 2 introduces messaging protocols with the help of two different ways of specifying them, one in terms of control flow and the other in terms of information flow. Section 3 introduces the notions of commitments and commitment protocols. Section 4 introduces the various kinds of reasoning one can do with protocol specifications. Section 5 discusses challenges in interaction protocol specification. Section 6 discusses some of the relevant literature.

2 Messaging Protocols

Languages for messaging protocols typically specify ordering and occurrence constraints between messages (more generally, message types). Typically these languages are operational in nature, meaning that the constraints are specified via control flow constructs such as sequence, choice, parallel, and so on, though there are also some declarative approaches [Montali *et al.*, 2010]. Important theoretical challenges here concern correct enactment in the face of distribution (general multiparty settings and asynchrony) and minimal message delivery assumptions. For example, we would not want to assume FIFO delivery of messages between agents; such an assumption would be inadequate anyway in settings of more than two agents. Correctness is usually expressed in terms of liveness (e.g., deadlock freedom) and safety properties (e.g., agents making compatible choices). Representations vary in formality and sophistication, ranging from Agent UML (AUML) [Odell *et al.*, 2000; Bauer *et al.*, 2001] and state machines to Petri Nets [El Fallah-Seghrouchni *et al.*, 2001], and pi-calculus. WS-CDL [WS-CDL, 2005] is a protocol language for modeling interactions among Web services. The Blindingly Simple Protocol Language (BSPL) is different from the above approaches for specifying protocols. Instead of specifying constraints directly between messages, in BSPL, one specifies constraints between the information items in a message.

Below we discuss AUML and BSPL as exemplar languages for specifying messaging protocols—AUML because it is based on UML notations that are widely used in software engineering, and BSPL because of (1) its basis in information, (2) its contrast with AUML, and (3) its support of fully decentralized asynchronous protocol enactments.

2.1 An Operational Approach: AUML

A AUML Sequence Diagram is an enhancement of a UML interaction diagram, which is an informal graphical notation for specifying interactions protocols. One specifies a protocol in AUML by specifying the control flow between messages using abstractions such as sequence, alternative, loop, and so on.

Figure 1 shows the FIPA (Foundation of Intelligent Physical Agents) Request Interaction Protocol in AUML. This protocol involves two roles, an INITIATOR and a PARTICIPANT. The INITIATOR sends a *request* to the PARTICIPANT, who either responds with a *refuse* or an *agree*. In the latter case, it follows up with a detailed response, which could be a *failure*, an *inform-done*, or an *inform-result*. The PARTICIPANT will omit the *agree* message unless the INITIATOR asked for a notification.

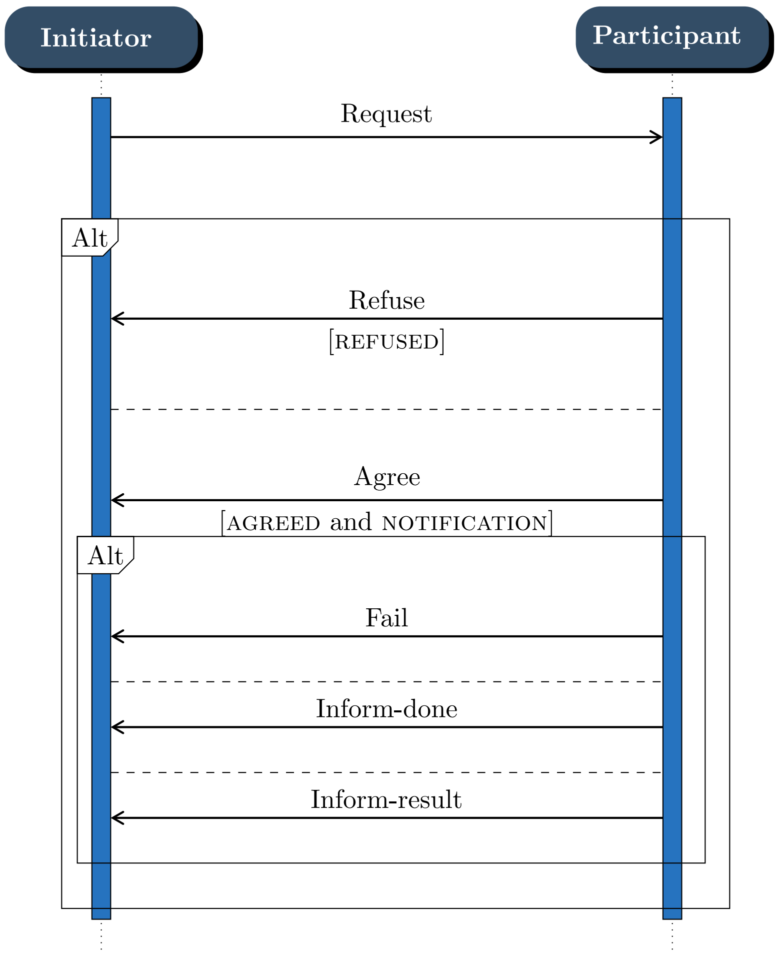

Figure 1. FIPA Request Interaction Protocol, from the FIPA specification [FIPA, 2003], expressed as a Agent UML (AUML) Sequence Diagram.

AUML sequence diagrams are a kind of automata which prescribes the legal sequencing of the protocol messages. The choice of relying on automata of some kind is well-supported in the literature concerning interaction protocols (see also [Cabac *et al.*, 2003; Dunn-Davies *et al.*, 2005]) but, as [Yolum and Singh, 2002; Winikoff *et al.*, 2004] point out, such protocol specifications show a rigidity that prevents agents from taking opportunities and handling exceptions in a dynamic and uncertain multiagent environment. This limitation can be overcome by commitment-based interaction protocols as we describe in Section 3.

2.2 Information-Oriented Approach: BSPL

The Blindingly Simple Protocol Language (BSPL) [Singh, 2011a] is a declarative, information-oriented approach for specifying protocols. Specifically, in a BSPL protocol specification, one eschews explicitly specifying ordering and occurrence relations between messages in favor of specifying the dependencies between the data communicated by the messages.

Listing 1.1. FIPA Request protocol in BSPL.

```
FIPA-Request {
  roles I, P // Initiator, Participant
  parameters out ID key, out job, out conf, out finished

  I ↦ P: Request[out ID, out job]
  P ↦ I: Agree[in ID, out conf]
  P ↦ I: Refuse[in ID, out conf, out finished]
  P ↦ I: Fail[in ID, in conf, out finished]
  P ↦ I: Done[in ID, in conf, out finished]
  P ↦ I: Result[in ID, in conf, out finished]
}
```

Listing 1.1 shows a protocol in BSPL that is analogous to the FIPA Request protocol. A BSPL protocol starts with declarations of the protocol's name, roles of the participants, and a set of protocol parameters. In Listing 1.1, the protocol's name is FIPA-Request. It includes the roles *I* (initiator) and *P* (participant), and the parameters *ID*, *job*, *conf*, and *finished*, from which *ID* is declared key. Hence, bindings of *ID* identify different enactments of the protocol. All protocol parameters are functionally dependent on the key parameters, which ensures integrity. A protocol enactment is said to be *complete* when all its parameters are bound. Each protocol parameter may be adorned out or in. In Listing 1.1, all protocol parameters are adorned out, meaning that their bindings are produced during the protocol's enactment. On the other hand, an in adorned parameter, which does not exist in Listing 1.1, indicates that the binding of that parameter must be provided externally. Overall, roles and parameters facilitate composition with other protocols. (BSPL also supports private roles and parameters, and an additional nil adornment, but we omit them here for brevity.)

The rest of Listing 1.1 declare the messages of the protocol, for which the declaration order is not significant. Each message declaration includes the sender and receiver, the name of the message, and its parameters. For example, *Request* is from *I* to *P*, and *ID* and *job* are its parameters. All key protocol parameters that appear in a message declaration are also key parameters for a message. Further, each message parameter is adorned out or in to capture causality between messages. For instance, *ID* and *job* are adorned out for *Request*, meaning that their bindings must be produced by *I* when sending *Request*. In *Agree*, *ID* and *job* are both in, meaning that their bindings must be known by *I* from the emission or reception of a prior message (e.g., reception of *Request*) to be able to send *Agree*. Beside causality, *out* adornment also ensures mutual exclusion. That is if a parameter is adorned *out* it more than one messages (e.g., *Agree* and *Refuse*), only one of these message can be sent in an enactment. Otherwise, the common parameter could have different bindings for the same key and violate integrity.

BSPL is formal and is more expressive than AUML. BSPL explicitly supports parameters and, through keys, the identification of protocol instances. BSPL is declarative and is designed to support composition. Intuitively, a message declaration is an atomic protocol; any other protocol (e.g. *FIPA Request*) is a composition of protocols. BSPL specifications may be verified for liveness and safety [Singh, 2012]. Implementations of BSPL in middleware can ensure that only correct messages (that preserve integrity and respect the information flow constraints) may be sent by any party [Singh, 2011b]. Winikoff et al. [2018] discuss the differences between AUML and BSPL in detail.

3 Commitment-Based Interaction Protocols

Commitment-based interaction protocols are meaning-based. They capture interaction patterns in a declarative way given in terms of commitments, involving a set of predefined roles. By executing protocol actions agents can create new commitments or manipulate existing commitments, e.g. they can release another agent from some commitment. Any agent who knows the meaning of the protocol actions, and is aware of a sequence of executions of such actions, will be able to deduce which commitments hold and their state. This section introduces social commitments and commitment-based interaction protocols, including both their basic definition (Section 3.1) and more recent evolutions that allow tackling temporal regulations inside commitments as well as dialectical commitments (Section 3.2).

3.1 Fundamentals of Social Commitments

Social Commitments. Since the late 1980s, many studies on distributed artificial intelligence, on formal theories of collective activity, on team work, and on cooperation [Castelfranchi, 1995; Singh, 1997; Norman *et al.*, 2004] implicitly identified *commitment*, the glue of group activity. Commitments link the actions of the group members and the group members with each other. The value of commitments as fundamental building blocks of interaction is recognized also outside artificial intelligence, like in sociology, where commitments are fundamental to the definition of organizations [Elder-Vass, 2010], and in economics where models based on commitment and trust were, for instance, proposed to understand the functioning of relational marketing [Morgan and Hunt, 1994].

This chapter relies on the notion of *social commitment*, as defined by Singh [1999]. *C(debtor, creditor, antecedent, consequent)* denotes a social commitment, meaning that the *debtor* debtor is committed to the *creditor* creditor to bring about the *consequent* consequent condition, when the antecedent *antecedent* condition holds. For instance, *C(merchant, customer, goods-paid, goods-delivered)* means that if the merchant is paid for goods, then merchant is committed to the customer to deliver the goods. Hence, commitments let agents have *expectations* on the behavior of their counterparts. In particular, the creditor of a detached commitment expects that its debtor will sooner or later bring about the consequent condition—debtors are, indeed, *liable* in case their commitments are violated.

With reference to [Telang *et al.*, 2012] each social commitment can be in one of the following states. A commitment is *Violated* when its antecedent is true but its consequent will forever be false, or it is canceled when *Detached*; *Satisfied*, meaning that the engagement is accomplished; *Expired*, meaning that it is no longer in effect and therefore the debtor would not fail to comply even if does not accomplish the consequent. Typically, a commitment should be *Active* when it is initially created. Active has two substates: *Conditional* (as long as the antecedent does not occur) and *Detached* (the antecedent has occurred).

It is typically assumed that only the debtor has control over the creation and discharge of commitments and the creditor has control over detachment. Further, the literature discusses commitment operations such as delegation (to a new debtor) and assignment (to a new creditor) that can only be done by the debtor and creditor, respectively [Singh, 1999].

Commitment-based interaction protocols. *Commitment protocols* were introduced in the seminal works by Yolum and Singh [2001a; 2001b]. Agents share a social state that contains commitments and other literals that are relevant to their interaction. Every agent can affect the social state by executing actions, whose definition is given in terms of updates to the social state (e.g. add a new commitment, release another agent from some commitment, satisfy a commitment). Hence, in its basic interpretation, a commitment protocol is made of a *set of actions*, involving the foreseen roles and whose semantics is agreed upon by all of the participants [Yolum and Singh, 2001a; Yolum and Singh, 2001b; Chopra, 2009]. The only constraint that commitment protocols include, to say that an interaction is successful, is that all commitments are discharged. These characteristics give commitment-based protocols great flexibility and give to the involved agents great autonomy. In fact, they are free to apply the social actions in any order they wish if, in the end, commitments are discharged.

Several languages have been developed in the recent years to specify commitment protocols (and other normative constructs). Listing 1.2 shows an example commitment protocol specification in the Custard language [Chopra and Singh, 2016a]. The protocol models the interaction between two agents, namely a *merchant* and a *customer*, in a purchase scenario. It involves a single commitment, namely *Purchase*, where the *merchant* is the debtor and the *customer* is the creditor. The state of the commitment is determined with respect to a set of events. It is created when a *Quote* event occurs. It becomes detached if the *Goods-Paid* event occurs within ten time units after *Quote*. Otherwise, the commitment expires. Finally, the commitment becomes discharged if the *Goods-Delivered* event occurs within five time units after *Goods-Paid*.

Listing 1.2. A commitment protocol in Custard.

```
commitment Purchase merchant to customer
 create Quote
 detach Goods–Paid[, Quote + 10]
 discharge Goods–Delivered[, Goods–Paid + 5]
```

The events of the protocol are defined in an information schema as we show in Listing 1.3, where each event corresponds to a relation. For instance, a *Quote* event has three attributes, namely, *purchaseID*, *goods*, and *price*, which capture the information associated with the event. Furthermore, the attribute *purchaseID* is defined *key* for this event. Instances of the events in a schema are stored in the information stores of the agents in a distributed manner. The event instances should satisfy the integrity constraints of the schema (e.g., each event instance must have a unique key). This information-oriented approach enables management of multiple protocol instances (e.g., each *Quote* event initiates a new protocol with a unique *purchaseID*). States of protocol instances can be determined by querying the information store, which is generated automatically by Custard from the protocol specification.

Listing 1.3. A commitment protocol in Custard.

```
schema
 Quote(purchaseID, goods, price) key purchaseID time t
 Goods–Paid(purchaseID, goods, paymentInfo) key purchaseID time t
 Goods–Delivered(purchaseID, goods, deliveryInfo) key purchaseID time t
```

Realistic scenarios often involves multiple agents and multiple commitments between them. For instance, in our purchase example there could be a third courier agent to realize delivery of goods. Listing 1.4 extends the previous protocol with a new commitment, namely *Delivery*, from the courier to the merchant to capture the delivery of goods. Intuitively, the commitment states that, if there is a delivery order by the merchant, the courier should do the delivery within three time units. We omit the extended event schema for the new events *Delivery-Agreement* and *Delivery-Order* for brevity.

Listing 1.4. A commitment protocol in Custard.

```
commitment Purchase merchant to customer
 create Quote
 detach Goods–Paid[, Quote + 10]
 discharge Goods–Delivered[, Goods–Paid + 5]

commitment Delivery courier to merchant
 create Delivery–Agreement
 detach Delivery–Order
 discharge Goods–Delivered[, Delivery–Order + 3]
```

It is straightforward to extend the above protocol further with additional agents and commitments. Listing 1.5 shows two more commitments that can be added to the above protocol. The first commitment, namely *Payment*, captures the interaction of the customer with a new *bank* agent to realize the payment. The second commitment, namely *Refund* extends the interaction of the customer and merchant for the refund of returned goods. Note that detachment of *Refund* occurs when *Purchase* is discharged, which is a meta-level commitment state event.

Listing 1.5. A commitment protocol in Custard.

```
// Purchase and Delivery commitments as defined above

commitment Payment bank to customer
 create Open-Account
 detach Payment-Order
 discharge Goods-Paid [, Payment-Order + 2]

commitment Refund merchant to customer
 create Quote
 detach discharged Purchase and Goods-Returned [, Goods-Delivered + 5]
 discharge Refunded [, Goods-Returned + 5]
```

These examples show several strengths of commitment protocols. First, commitment protocols support autonomy of agents. That is, a commitment protocol defines the expectations of agents, but it does not define how these expectations should be satisfied. Hence, agents are free to fulfill (or not) these expectations as they see fit. For instance, the customer could realize the payment either herself or through the bank. Similarly, the merchant could herself deliver the goods or make a delivery order to the courier. Note that a procedural specification approach, such as AUML, can also represent such situations. However, in most cases, the number of possible realizations of an interaction grows rapidly and makes the procedural approach impractical. Second, commitment protocols are modular, which makes them easy to extend with new agents and commitments, and also compose with existing protocols as our examples demonstrate.

3.2 Advanced Concepts

Temporal Regulations inside Commitment Protocols. Some works explored the possibility to introduce temporal regulations inside commitment protocols to capture constraints on the coordination between agents. 2CL [Baldoni *et al.*, 2013] is a first proposal that aims capturing temporal regulations on the flow of events in commitment-based protocols, decoupling *constitutive* rules from *regulative* rules. Following Searle [1995], constitutive rules, by identifying certain behaviors as foundational of a certain type of activity, create that activity. Regulative rules, instead, contingently constrain a previously constituted activity. The decoupling of constitutive and regulative components is motivated by desired properties such easier re-use of actions in different contexts, easier customization of protocols, and easier composition of protocols. A multi-agent system that clearly separates a constitutive component from a regulative component in its protocols gains greater openness, interoperability, and modularity of design.

While 2CL enriches commitment-based protocols with temporal regulations an agent is said to accept when enacting a protocol role, the proposal of Marengo et al. [2011] improves the idea by giving regulations a real normative power. It does so by allowing the specification of commitments to temporal regulations. Consider the following example. A physician commits to a patient that if the patient has any sign of heart trouble after signing up with him, then the patient will be immediately referred to a laboratory for tests, the results of which will be evaluated by a specialist. Temporal constraints such as those in the example are traditionally captured as procedural workflows. Marengo *et al.* capture such constraints more broadly as regulations and express them more flexibly in a logical notation. The commitments among autonomous parties capture their business relationships naturally. Regulations are incorporated into the contents of commitments. Thus by reifying regulations into business relationships, normative force is brought to the specification, thereby providing a clear basis for the participants to guide their actions locally and to judge the compliance of their counterparties.

Commitment antecedent and consequent conditions are given in precedence logic. Precedence logic is an event-based logic [Singh, 2003]. It has three primary operators for specifying requirements about the occurrence of events: '$\vee$' (choice), '$\wedge$' (concurrence), and '$\cdot$' (before). The *before* operator enables one to express specifications such as $\mathit{approve} \cdot \mathit{perform}$: both *approve* and *perform* must occur and in the specified order. The specifications are interpreted over runs. Each run is a sequence of events. The transitions correspond to event occurrences.

However, placing regulations inside commitments, as in the above example, leads to new challenges, among which the need to formalize the progression (the life cycle) of commitments, bearing in mind the events that have occurred. For example, we say that an active commitment $C(x, y, r, u)$ progresses to *satisfied* when u occurs. Analogously, we would like to say that $C(x, y, r, e \cdot f)$ progresses to $C(x, y, r, f)$ when e occurs. The challenge is to formalize general progression rules for an expressive event language.

Type Checking. While a main research issue in concurrency programming, including the actor model [Agha, 1986], is the definition of formal models, and type systems analogously to the sequential case, for what concerns agents, just a few studies were carried on aspects such as typing and type safety. In programming languages, type systems are used to help designers and developers to avoid code errors that can produce unpredictable results. In MAS, and in particular for what concerns interaction inside a MAS, typing systems would provide a means for answering questions like: does an agent have the means for carrying out the encoded interaction? Is an agent compliant to a role specification?

Baldoni et al. [2014a; 2018] propose an agent-based, dynamic (i.e., performed at role enactment time), and declarative type checking system for agent interactions that are modeled by means of commitment-based protocols. The typing system enjoys an important progression property: when an agent joins an interaction, it is guaranteed that it possesses all the required behaviors to carry out its part, as far as one of the interaction acceptance states. One of the reasons to express agent types and role requirements based on commitments is that the agent system developers are better supported if the typing system relies on abstractions that are typical of MASs, rather than relying on abstractions from other programming paradigms, as done by other proposals. Clearly, a type system allows only a lightweight check of the behavior of the involved agents, being more concerned with a safe usage rather than a full behavioral compatibility. It does not imply that an agent which has the same type of another agent will display the same behavior. This does not exclude the possibility to integrate deeper checks, for instance based on model checking such as [Bentahar *et al.*, 2009]. Type checking, as a form of lightweight verification, adopts notions (e.g., substitutability), that are used also for coping with issues of interoperability and conformance, discussed in [Baldoni *et al.*, 2006; Baldoni *et al.*, 2009]. The conformance verification aims at guaranteeing that when an agent plays a role, or substitutes another agent in an on-going interaction, the interoperability of the system is preserved.

The explained commitment-based typing proposal overcomes the limitations of approaches to the typing of interaction protocols that are based on global session types by Ancona et al. [2013]. Briefly, Ancona et al. use global session types as an abstraction tool, which enables automatically generating monitor agents. A monitor agent has the aim of verifying the correctness of a multiparty interaction; to this aim, it intercepts all the exchanged messages and verifies whether the protocol is respected. Similarly to finite state automata, global session types have a prescriptive, procedural nature, whose drawbacks have been explained [Yolum and Singh, 2002; Winikoff *et al.*, 2004]. Moreover, as the reader will observe, in this proposal typing is used to specify the interaction within a system from a global perspective, rather than provide an actual typing of the agents. Thus, questions concerning whether an agent can enact a role, or has some abilities required by the interaction remain unanswered.

4 Reasoning over Commitments

Commitment-based interaction protocols enable agents to reason about their interactions with others without knowing their internals. There are several reasoning tasks that should be performed by an agent to ensure its effective operation when enacting a protocol. Here we identify three such tasks: (i) whether a protocol *supports* an agent to achieve its goals, (ii) whether a protocol is *feasible* for an agent to enact, and (iii) whether a protocol is *verifiable.*

4.1 Goal Support

goal support Goal support [Günay *et al.*, 2013; Günay *et al.*, 2015b] generalizes the ideas of control and safety to ensure that an agent achieves its goals with respect to its commitments. That is, a protocol supports the goals of an agent, if the commitments of the protocol provide sufficient control to the agent for achieving its goals. For instance, suppose that a protocol involves the commitment $C(merchant, customer, goods\text{-}paid, goods\text{-}delivered)$, and the goal of the customer is to achieve *goods-delivered*. If we assume that the merchant fulfills its commitments, we can say that the commitment supports the customer's goal, since it makes the merchant committed for delivering the goods. However, note that the antecedent of the commitment requires a payment to occur first. Hence, the customer should adopt an intermediary goal to bring about *goods-paid* [Telang *et al.*, 2012], and the protocol should also support this intermediary goal. If the customer controls the payment, we can conclude that the customer's goal is supported by the above commitment, since no matter what happens, the customer can make a payment and the merchant becomes committed to deliver the goods. If the customer does not control payment, the protocol supports the customer's goal, only if there is another commitment in the protocol that ensures occurrence of the payment. Kafalı et al. [2014] address computation of goal support using event calculus. Günay et al. [2015a; 2016] extend computation of goal support into a probabilistic setting considering uncertainty in the behaviors of agents.

4.2 Feasibility

feasibility In many settings agents need resources to perform actions. In this context, a commitment protocol is feasible for an agent, if either the agent has sufficient resources before the enactment of the protocol to fulfill its commitments, or the agent can acquire the necessary resources from other agents during the enactment of the protocol, which is subject to the commitments of the protocol and their feasibility [Günay and Yolum, 2013]. For instance, a merchant's commitment to deliver certain goods is feasible, if the goods are already in the merchant's stock. If this is not the case, there should be other feasible commitments which ensure acquisition of the goods by the merchant (e.g., a commitment from a supplier to provide the goods). Note that the definition of feasibility is recursive (e.g., the merchant should have sufficient resources to detach the supplier's commitment). Time constraints (e.g., deadlines) should also be taken into account for feasibility. If the merchant was committed to deliver the goods in three days, but the supplier is committed to provide the goods in five days, than the merchant's commitment might not be feasible. Günay and Yolum [2011; 2013] use constraint satisfaction techniques to reason about feasibility of a commitment protocol for an agent.

4.3 Verifiability

verifiability We have seen that it is possible, by applying proper kinds of reasoning, to assess whether playing a protocol role is safe for an agent, and whether the agent has sufficient control. A related property concerns whether agents can or cannot judge the compliance, to the agreed protocol, of their parties in the interaction, as the interaction proceeds. This problem is studied in [Baldoni *et al.*, 2015b], where an agent is said to be *compliant* if and only if it discharges all commitments of which it is the debtor. The research question is to characterize commitment protocols that, when enacted, support each participant verifying the compliance of the other participants. Verification requires each participant to be in condition to observe "relevant" events so that it can determine the progress of the commitment along its lifecycle. The relevant events are all those that affect commitments, e.g. by creating, detaching, violating them. An interaction is *verifiable* when relevant events can be observed by all participants. Thus, each of them knows which commitments are active (or expired, detached, or violated), and consequently knows about its own and the other principal's compliance. The problem is that, as in general, not all the concerned parties can observe all of the events relevant to their commitments. A simple example is that of a purchase, where delivery is made by a courier. As such, its completion is not directly observable by the seller who, however, is the one who has a commitment towards the client for delivery. So, for instance, the seller will not know, by direct observation, that the client received the goods (event which satisfies the commitment). The client's and the seller's views of the state of the interaction cannot be "aligned" alignment [Chopra and Singh, 2009; Chopra and Singh, 2015], hence yielding non-verifiability.

A solution is suggested by the way in which similar situations are tackled in the real world. When relevant events cannot be observed, the participants of the interaction enrich the protocol with new actions (like delivery tracking), whose meaning is a claim about the state of affairs (a position). Such claims imply the taking of responsibility of the truth of what is declared. So, the courier will let the seller know the state of the delivery by declaring it through a tracking service. Claims act as "bridges" between different interaction contexts. Baldoni et al. [2015b] observe that such claims raise expectations by some principals of others, and that each principal is accountable for the expectations it creates in others. Consequently, they propose to rely on *dialectical* commitments [Singh, 2008] to represent agents' claims. They also introduce a pragmatic rule because of which the participants can use dialectical commitments as equal to the conditions they concern. In other words, a dialectical commitment by the courier saying delivery occurred will satisfy the seller's commitment to delivery. By realizing claims as commitments, debtors become *accountable* for their declarations, and are held *liable* when the commitment is violated.

5 Challenges

Although foundations of commitment-based protocols are well studied, there are still several challenges to address to realize their use in practice. Below, we discuss two of these challenges in detail.

5.1 Run-time Protocol Synthesis

Typically, commitment protocols are defined at design-time, and embedded into the implementations of agents [Winikoff, 2007]. This approach simplified the development of a MAS. However, it also limits flexibility and adaptation of a MAS, since it requires protocols and agents to be tightly coupled. The alternative of design-time protocols is to enable agents in a MAS to create their own protocols at run-time according to their needs [Artikis, 2009]. synthesis We call this approach *run-time protocol synthesis*. The specific problem of run-time protocol synthesis is the following [Günay *et al.*, 2015b]: given a set of agents, their goals, preferences, and capabilities, automatically synthesize a commitment protocol to regulate the interactions of the agents, which enables the agents to achieve their goals with respect to their preferences and capabilities. This problem is relatively straightforward to solve (at least conceptually) in a centralized manner for a simple propositional representation. However, realistic applications (e.g., IoT, healthcare) need a decentralized synthesis approach that supports an expressive representation, such as first-order logic. Decentralization is essential for efficiency and reliability. Besides, it is also necessary to ensure privacy of the agents. That is, the synthesis mechanism should not force agents to reveal their goals, preferences, and capabilities to others.

There are two recent approaches that address run-time synthesis of protocols. Telang et al. [2013; 2013] develop a centralized planning approach that uses hierarchical task network (HTN) planning to synthesize commitment protocols. Günay et al. [2013; 2015b] develop a distributed approach for run-time synthesis of protocols that is based on the goal support property we have discussed earlier. These two proposals establish the foundations to solve the run-time protocol synthesis challenge. However, both methods have some shortcomings. Telang et al. achieve a desirable degree of expressiveness by using a first-order formalization. However, they consider a fully centralized setting, in which all the information about the agents (e.g., private goals) are known to a central planner. On the other hand, Günay et al. do not require agents to share their private information. However, they use a propositional formalization, which suffers from limited expressive power. Besides, neither of these methods consider constraints such as time and resources. In the light of our discussion and the proposals, we identify the properties of an ideal run-time protocol synthesis method as follows: (i) The run-time protocol synthesis problem should be solved in a fully automated manner at run-time by the agents without human intervention. (ii) The synthesis method should be capable of creating commitment protocols in a sufficiently expressive representation to capture various aspects of practical problems, such as fine-grained time requirements and resource constraints. (iii) The method should be decentralized and preserve privacy of agents. There is a significant body of work both in multiagent planning [Durfee and Zilberstein, 2013] and distributed constraint optimization [Yokoo *et al.*, 1998] techniques. We believe that utilization of these techniques is a promising direction for the solution of run-time protocol synthesis challenge.

5.2 Methodologies and Tools

Effective methodologies and tools for designing commitment protocols are required to integrate commitment protocols into the engineering processes of MAS. There are several proposals in this direction. Winikoff [2006; 2007] discusses implementation of commitment protocols in conventional agent-oriented programming languages, and develops a mapping from commitment protocols to BDI-style plans with extensions to semantics of BDI languages. Yolum [2007] formalizes design requirements of commitment protocols, such as effectiveness, consistency, and robustness and develops tools to analyze commitment protocols with respect to these requirements. Amoeba [Desai *et al.*, 2009] is a methodology for designing business processes in which commitment protocols capture the business meaning of interactions among autonomous agents. Amoeba describes guidelines to specify cross-organizational processes using commitment protocols and uses composition to handle the evolution of requirements. 2CL Methodology [Baldoni *et al.*, 2014b] extends Amoeba with temporal constraints. JaCaMo+ [Baldoni *et al.*, 2015a] is a programming framework where Jason agents interact according to commitment-based protocols, which are modeled as special CArtAgO artifacts. JaCaMo+ handles enactment and monitoring of commitment-based protocols, including the social state of the interaction, and enables Jason agents both to be notified about the social events and to perform practical reasoning about the other agents' actions. Torroni and colleagues [Chesani *et al.*, 2013] address several issues about monitoring of commitment protocols including exceptions [Kafalı and Torroni, 2012] and delegations [Kafalı and Torroni, 2011], and develop tools using reactive event calculus. Kafalı and Yolum [2016] also develop a methodology to create agents that can monitor interactions of their users in business settings. They use a flexible as-good-as relation to compare the state of the interaction with the expectations of the users.

Despite these methodologies and tools, there are still several issues to address for better integration of commitment protocols into the engineering of practical MAS. First, there is no standardization on the representation and interpretation of commitments. While all the methodologies and tools share a common basic model of commitments, they use different approaches to handle details such as representation of lifecycle, information, and time. Hence, development of standard models of commitments is necessary for the interoperability of different methodologies and tools. Second, most methodologies and tools are general purpose and inadequate to address domain specific issues of modern systems such Internet of Things and healthcare. Finally, there is no formal approach for the operationalization of commitment protocols (i.e., mapping of a commitment protocol to operational messages), which is essential for the efficient development of correct commitment-based systems, by eliminating manual transformations from commitment protocols to operational messages.

6 Related Approaches

In the literature there different proposals for specifying meaning-based interaction protocols. An approach that lies between automata-based approaches (like AUML [Bauer *et al.*, 2001]) and commitment protocols is the one proposed in [Fornara, 2003; Fornara and Colombetti, 2004] which provides a commitment-based semantics for the speech acts of an agent communication language, and relies on interaction diagrams to regulate the interaction.

Proposals concerning Electronic Institutions, like the seminal papers by Esteva et al. [2004] and Tinnemeier et al. [2009], take a different perspective. Esteva et al. use an interaction protocol to regulate the dialogic interactions of a group of agents, constituting a scene. Scenes can be composed resulting into complex interactions with transitions from a scene to another. *Normative meaning* is associated to the actions (in [Tinnemeier *et al.*, 2009] to facts) leading to the generation of obligations (prohibitions, etc.) that will then shape the interaction within the MAS. Once again in this kind of work, protocols often amount to automata prescribing specific courses of interaction.

Torroni et al. [2009] propose the idea of relying on *social expectations* as an alternative to commitments for providing a social semantics to the interaction. An expectation represents a desired behavior by describing an event and the time at which the event is expected to happen. Positive expectations have the form $E(p,t)$, meaning that the event p is expected to happen at time t. Negative expectations have the form $EN(p,t)$ capturing that event p should not happen at time t. A main difference of expectations from commitments is that expectations are not directed from a debtor to a creditor. Therefore, even if they have a normative aspect, expectations are not associated to a notion of accountability. This approach is at the basis of the SCIFF logical framework [Alberti *et al.*, 2008] that supports both interaction specification and verification. A SCIFF specification is an abductive logic program, where social integrity constraints are used to model relations among events as well as expectations about events. Based on such integrity constraints, it is possible to define patterns of interactions by relating the time at which different events are expected to happen.

Interaction protocols bear similarities with business processes. In recent years, as Chesani et al. [2009] reports, workflow management systems for business processes address trade-offs between flexibility and expressiveness, introducing ways for deviating from the standardized process and even modifying the process. Verifying the compliance of complex and flexible processes to regulations is a hard task that cannot be performed a priori, due to the fact that the involved agents can stray from the original process or even modify it. Compliance verification, in such cases, is usually performed afterwards. An analyst observes traces of (possibly modified) process executions to understand if any violation occurred. The expectation-based approach was used to implement a tool that performs this kind of check in an automatic way.

The extensive work on interactions has led Chopra and Singh [Chopra and Singh, 2016b] to formulate Interaction-Oriented Software Engineering (IOSE) as a distinct paradigm from Agent-Oriented Software Engineering (AOSE). Whereas in AOSE, the focus is typically on specifying and composing agents and interaction is seen as instrumental to achieving the goals of agents, in IOSE the focus is on the specification and composition of interaction protocols without reference to agents. This separation of agent concerns from interaction concerns is likely to be of crucial importance going forward.

Acknowledgments: We thank Michael Winikoff and Neil Yorke-Smith for their extensive comments on an earlier draft. Chopra and Günay were supported by EPSRC grant EP/N027965/1 (Turtles).

BIBLIOGRAPHY

[Agha, 1986] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA, USA, 1986.

[Alberti *et al.*, 2008] Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni. Verifiable agent interaction in abductive logic programming: The SCIFF framework. *ACM Transactions on Compututational Logic*, 9(4):29:1–29:43, 2008.

[Ancona *et al.*, 2013] Davide Ancona, Sophia Drossopoulou, and Viviana Mascardi. Automatic generation of self-monitoring MASs from multiparty global session types in Jason. In *Declarative Agent Languages and Technologies X*, volume 7784 of *LNCS*, pages 76–95. Springer Berlin Heidelberg, 2013.

[Artikis, 2009] Alexander Artikis. Dynamic protocols for open agent systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 97–104, 2009.

[Baldoni *et al.*, 2006] Matteo Baldoni, Cristina Baroglio, Alberto Martelli, and Viviana Patti. A priori conformance verification for guaranteeing interoperability in open environments. In *Service-Oriented Computing – ICSOC 2006*, volume 4294 of *LNCS*, pages 339–351. Springer Berlin Heidelberg, 2006.

[Baldoni *et al.*, 2009] Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, Nirmit Desai, Viviana Patti, and Munindar P. Singh. Choice, interoperability, and conformance in interaction protocols and service choreographies. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 843–850, 2009.

[Baldoni *et al.*, 2013] Matteo Baldoni, Cristina Baroglio, Elisa Marengo, and Viviana Patti. Constitutive and regulative specifications of commitment protocols: a decoupled approach. *ACM Transactions on Intelligent Systems and Technology, Special Issue on Agent Communication*, 4(2):22:1–22:25, 2013.

[Baldoni *et al.*, 2014a] Matteo Baldoni, Cristina Baroglio, and Federico Capuzzimati. Typing multi-agent systems via commitments. In *Engineering Multi-Agent Systems*, volume 8758 of *LNCS*, pages 388–405. Springer Berlin Heidelberg, 2014.

[Baldoni *et al.*, 2014b] Matteo Baldoni, Cristina Baroglio, Elisa Marengo, Viviana Patti, and Federico Capuzzimati. Engineering commitment-based business protocols with the 2CL methodology. *Autonomous Agents and Multi-Agent Systems*, 28(4):519–557, 2014.

[Baldoni *et al.*, 2015a] Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. Leveraging commitments and goals in agent interaction. In *Proceedings of the 30th Italian Conference on Computational Logic*, volume 1459 of *CEUR Workshop Proceedings*, pages 85–100. CEUR-WS.org, 2015.

[Baldoni *et al.*, 2015b] Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, and Munindar P. Singh. Composing and verifying commitment-based multiagent protocols. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 10–17, 2015.

[Baldoni *et al.*, 2018] Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. Type checking for protocol role enactments via commitments. *Journal of Autonomous Agents and Multi-Agent Systems*, 32(3):349–386, 2018.

[Bauer *et al.*, 2001] Bernhard Bauer, Jörg P. Müller, and James Odell. Agent UML: A formalism for specifying multiagent software systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):207–230, 2001.

[Bentahar *et al.*, 2009] Jamal Bentahar, John-Jules Ch. Meyer, and Wei Wan. Model checking communicative agent-based systems. *Knowledge-Based Systems*, 22(3):142–159, 2009.

[Bravetti and Zavattaro, 2009] Mario Bravetti and Gianluigi Zavattaro. A theory of contracts for strong service compliance. *Mathematical Structures in Computer Science*, 19(3):601–638, 2009.

[Cabac *et al.*, 2003] Lawrence Cabac, Daniel Moldt, and Heiko Rölke. A proposal for structuring petri net-based agent interaction protocols. In *Applications and Theory of Petri Nets 2003*, volume 2679 of *LNCS*, pages 102–120. Springer Berlin Heidelberg, 2003.

[Castelfranchi, 1995] Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In *Proceedings of the 1st International Conference on Multiagent Systems*, pages 41–48, 1995.

[Chesani *et al.*, 2009] Federico Chesani, Paola Mello, Marco Montali, Fabrizio Riguzzi, Maurizio Sebastianis, and Sergio Storari. Checking compliance of execution traces to business rules. In *Business Process Management Workshops*, volume 17 of *Lecture Notes in Business Information Processing*, pages 134–145. Springer Berlin Heidelberg, 2009.

[Chesani *et al.*, 2013] Federico Chesani, Paola Mello, Marco Montali, and Paolo Torroni. Representing and monitoring social commitments using the event calculus. *Autonomous Agents and Multi-Agent Systems*, 27(1):85–130, 2013.

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

[Chopra and Singh, 2015] Amit K. Chopra and Munindar P. Singh. Generalized commitment alignment. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, pages 453–461, 2015.

[Chopra and Singh, 2016a] Amit K. Chopra and Munindar P. Singh. Custard: Computing norm states over information stores. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*, pages 1096–1105, 2016.

[Chopra and Singh, 2016b] Amit K. Chopra and Munindar P. Singh. From social machines to social protocols: Software engineering foundations for sociotechnical systems. In *Proceedings of the 25th International World Wide Web Conference*, pages 903–914, 2016.

[Chopra, 2009] Amit K. Chopra. *Commitment Alignment: Semantics, Patterns, and Decision Procedures for Distributed Computing.* PhD thesis, North Carolina State University, Raleigh, NC, 2009.

[Desai *et al.*, 2009] Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Amoeba: A methodology for modeling and evolving cross-organizational business processes. *ACM Transactions on Software Engineering Methodolgy*, 19(2), 2009.

[Dunn-Davies *et al.*, 2005] H. R. Dunn-Davies, R. J. Cunningham, and S. Paurobally. Propositional statecharts for agent interaction protocols. *Electronic Notes in Theoretical Computer Science*, 134:55–75, 2005.

[Durfee and Zilberstein, 2013] Ed Durfee and Shlomo Zilberstein. Multiagent planning, control, and execution. In Gerhard Weiss, editor, *Multiagent Systems*, chapter 11, pages 485–546. MIT Press, 2013.

[El Fallah-Seghrouchni *et al.*, 2001] Amal El Fallah-Seghrouchni, Serge Haddad, and Hamza Mazouzi. A formal study of interactions in multi-agent systems. *International Journal of Computers and Their Applications*, 8(1), 2001.

[El-Menshawy *et al.*, 2011] Mohamed El-Menshawy, Jamal Bentahar, and Rachida Dssouli. Model checking commitment protocols. In *Modern Approaches in Applied Intelligence*, volume 6704 of *LNCS*, pages 37–47. Springer Berlin Heidelberg, 2011.

[Elder-Vass, 2010] Dave Elder-Vass. *The causal power of social structures: emergence, structure and agency*. Cambridge Univ Press, 2010.

[Esteva *et al.*, 2004] Marc Esteva, Bruno Rosell, Juan A. Rodríguez-Aguilar, and Josep Lluís Arcos. AMELI: an agent-based middleware for electronic institutions. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 236–243, 2004.

[FIPA, 2003] FIPA. FIPA interaction protocol specifications, 2003. FIPA: The Foundation for Intelligent Physical Agents.

[Fornara and Colombetti, 2004] Nicoletta Fornara and Marco Colombetti. A commitment-based approach to agent communication. *Applied Artificial Intelligence*, 18(9-10):853–866, 2004.

[Fornara, 2003] Nicoletta Fornara. *Interaction and Communication among Autonomous Agents in Multiagent Systems*. PhD thesis, Università della Svizzera italiana, Facoltà di Scienze della Comunicazione, 2003.

[Günay and Yolum, 2011] Akın Günay and Pınar Yolum. Detecting conflicts in commitments. In *Declarative Agent Languages and Technologies IX*, volume 7169 of *LNCS*, pages 51–66. Springer Berlin Heidelberg, 2011.

[Günay and Yolum, 2013] Akın Günay and Pınar Yolum. Constraint satisfaction as a tool for modeling and checking feasibility of multiagent commitments. *Applied Intelligence*, 39(3):489–509, 2013.

[Günay *et al.*, 2013] Akın Günay, Michael Winikoff, and Pınar Yolum. Commitment protocol generation. In *Declarative Agent Languages and Technologies X*, volume 7784 of *LNCS*, pages 136–152. Springer Berlin Heidelberg, 2013.

[Günay *et al.*, 2015a] Akın Günay, Song Songzheng, Yang Liu, and Jie Zhang. Automated analysis of commitment protocols using probabilistic model checking. In *Proceedinds of the 29th AAAI Conference on Artificial Intelligence*, pages 2060–2066, 2015.

[Günay *et al.*, 2015b] Akın Günay, Michael Winikoff, and Pınar Yolum. Dynamically generated commitment protocols in open systems. *Autonomous Agents and Multi-Agent Systems*, 29(2):192–229, 2015.

[Günay *et al.*, 2016] Akın Günay, Yang Liu, and Jie Zhang. Promoca: Probabilistic modeling and analysis of agents in commitment protocols. *Journal of Artificial Intelligence Research*, 57(1):465–508, 2016.

[HL7, 2002] HL7. HL7 reference information model, version 1.19. http://www.hl7.org/Library/data-model/RIM/C30119/Graphics/RIM_billboard.pdf, 2002.

[Huget, 2004] Marc-Philippe Huget. Agent UML notation for multiagent system design. *IEEE Internet Computing*, 8(4):63–71, 2004.

[Kafalı and Torroni, 2011] Özgür Kafalı and Paolo Torroni. Social commitment delegation and monitoring. In *Computational Logic in Multi-Agent Systems*, volume 6814 of *LNCS*, pages 171–189. Springer Berlin Heidelberg, 2011.

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

[Kafalı and Yolum, 2016] Özgür Kafalı and Pınar Yolum. PISAGOR: A proactive software agent for monitoring interactions. *Knowledge Information Systems*, 47(1):215–239, 2016.

[Kafalı *et al.*, 2014] Özgür Kafalı, Akın Günay, and Pınar Yolum. GOSU: Computing goal support with commitments in multiagent systems. In *Proceedinds of the 21st European Conference on Artificial Intelligence*, pages 477–482, 2014.

[Marengo *et al.*, 2011] Elisa Marengo, Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, Viviana Patti, and Munindar P. Singh. Commitments with regulations: reasoning about safety and control in REGULA. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 467–474, 2011.

[Meneguzzi *et al.*, 2013] Felipe Meneguzzi, Pankaj R. Telang, and Munindar P. Singh. A first-order formalization of commitments and goals for planning. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, AAAI, pages 697–703, 2013.

[Montali *et al.*, 2010] Marco Montali, Maja Pesic, Wil M. P. van der Aalst, Federico Chesani, Paola Mello, and Sergio Storari. Declarative specification and verification of service choreographies. *ACM Transactions on the Web*, 4(1):3:1–3:62, 2010.

[Morgan and Hunt, 1994] Robert M. Morgan and Shelby D. Hunt. The commitment-trust theory of relationship marketing. *Journal of Marketing*, 58(3):20–38, 1994.

[Norman *et al.*, 2004] Timothy J. Norman, D. V. Carbogim, Eric C. W. Krabbe, and C. Douglas Walton. Argument and multi-agent systems. In *Argumentation Machines: New Frontiers in Argument and Computation*, pages 15–54. Springer Netherlands, 2004.

[Odell *et al.*, 2000] James Odell, H. Van Dyke Parunak, and Bernhard Bauer. Representing agent interaction protocols in UML. In *Agent-Oriented Software Engineering*, volume 1957 of *LNCS*, pages 121–140. Springer Berlin Heidelberg, 2000.

[Rajamani and Rehof, 2002] Sriram K. Rajamani and Jakob Rehof. Conformance checking for models of asynchronous message passing software. In *Computer Aided Verification*, volume 2404 of *LNCS*, pages 166–179. Springer Berlin Heidelberg, 2002.

[RosettaNet, 1998] RosettaNet. http://www.rosettanet.org, 1998.

[Searle, 1995] John R. Searle. *The construction of social reality.* Free Press, New York, NY, US, 1995.

[Singh, 1997] Munindar P. Singh. Commitments among autonomous agents in information-rich environments. In *Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent Rationality*, pages 141–155, 1997.

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

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

[Singh, 2003] Munindar P. Singh. Distributed enactment of multiagent workflows: temporal logic for web service composition. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 907–914, 2003.

[Singh, 2008] Munindar P. Singh. Semantical considerations on dialectical and practical commitments. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 176–181, 2008.

[Singh, 2011a] Munindar P. Singh. Information-driven interaction-oriented programming: BSPL, the blindingly simple protocol language. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems*, pages 491–498, 2011.

[Singh, 2011b] Munindar P. Singh. LoST: Local state transfer—an architectural style for the distributed enactment of business protocols. In *Proceedings of the 9th IEEE International Conference on Web Services*, pages 57–64, 2011.

[Singh, 2012] Munindar P. Singh. Semantics and verification of information-based protocols. In *Proceedings of the 11th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1149–1156, 2012.

[Telang *et al.*, 2012] Pankaj R. Telang, Munindar P. Singh, and Neil Yorke-Smith. Relating goal and commitment semantics. In *Programming Multi-Agent Systems*, LNCS, pages 22–37. Springer Berlin Heidelberg, 2012.

[Telang *et al.*, 2013] Pankaj R. Telang, Felipe Meneguzzi, and Munindar P. Singh. Hierarchical planning about goals and commitments. In *Proceedings of the 12th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 877–884, 2013.

[Tinnemeier *et al.*, 2009] Nick A. M. Tinnemeier, Mehdi Dastani, John-Jules Ch. Meyer, and Leendert W. N. van der Torre. Programming normative artifacts with declarative obligations and prohibitions. In *Proceedings of the 2009 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 145–152, 2009.

[Torroni *et al.*, 2009] Paolo Torroni, Federico Chesani, Pınar Yolum, Marco Gavanelli, Munindar P. Singh, Evelina Lamma, Marco Alberti, and Paola Mello. *Modelling Interactions via Commitments and Expectations*, chapter 11, pages 263–284. IGI Global, 2009.

[TWIST, 2006] TWIST. Transaction workflow innovation standards team, February 2006. http://www.twiststandards.org.

[Winikoff *et al.*, 2004] Michael Winikoff, Wei Liu, and James Harland. Enhancing commitment machines. In *Declarative Agent Languages and Technologies II*, volume 3476 of *LNCS*, pages 198–220. Springer Berlin Heidelberg, 2004.

[Winikoff *et al.*, 2018] Michael Winikoff, Nitin Yadav, and Lin Padgham. A new hierarchical agent protocol notation. *Autonomous Agents and Multi-Agent Systems*, 32(1):59–133, 2018.

[Winikoff, 2006] Michael Winikoff. Implementing flexible and robust agent interactions using distributed commitment machines. *Multiagent and Grid Systems*, 2(4):365–381, 2006.

[Winikoff, 2007] Michael Winikoff. Implementing commitment-based interactions. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 128:1–128:8, 2007.

[WS-CDL, 2005] WS-CDL. Web services choreography description language version 1.0, November 2005. www.w3.org/TR/ws-cdl-10/.

[Yokoo *et al.*, 1998] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 10(5):673–685, 1998.

[Yolum and Singh, 2001a] Pınar Yolum and Munindar P. Singh. Commitment machines. In *Intelligent Agents VIII*, volume 2333 of *LNCS*, pages 235–247, 2001.

[Yolum and Singh, 2001b] Pınar Yolum and Munindar P. Singh. Designing and executing protocols using the event calculus. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 27–28, 2001.

[Yolum and Singh, 2002] Pınar Yolum and Munindar P. Singh. Flexible protocol specification and execution: applying event calculus planning using commitments. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 527–534, 2002.

[Yolum, 2007] Pınar Yolum. Design time analysis of multiagent protocols. *Data and Knowledge Engineering*, 63(1):137–154, 2007.

Matteo Baldoni
Università degli Studi di Torino
Dipartimento di Informatica
via Pessinetto 12, 10149, Torino (TO), Italy
Email: matteo.baldoni@unito.it

Cristina Baroglio
Università degli Studi di Torino
Dipartimento di Informatica
via Pessinetto 12, 10149, Torino (TO), Italy
Email: cristina.baroglio@unito.it

Amit K. Chopra
Lancaster University
School of Computing and Communications
Lancaster, LA1 4WA, United Kingdom
Email: amit.chopra@lancaster.ac.uk

Akın Günay
Lancaster University
School of Computing and Communications
Lancaster, LA1 4WA, United Kingdom
Email: a.gunay@lancaster.ac.uk