# Correctness Properties for Multiagent Systems

Munindar P. Singh[1] and Amit K. Chopra[2]

[1] North Carolina State University, Raleigh, USA
singh@ncsu.edu
[2] Università degli Studi di Trento, Trento, Italy
akchopra.mail@gmail.com

**Abstract.** What distinguishes multiagent systems from other software systems is their emphasis on the interactions among autonomous, heterogeneous agents. This paper motivates and characterizes correctness properties for multiagent systems. These properties are centered on commitments, and capture correctness at a high level. In contrast to existing approaches, commitments underlie key correctness primitives understood in terms of meaning; for example, commitment *alignment* maps to *interoperability*; commitment *discharge* maps to *compliance*. This paper gives illustrative examples and characterizations of these and other properties. The properties cover the specification of the principal artifacts—protocols, roles, and agents—of an interaction-based approach to designing multiagent systems, and thus provide the formal underpinnings of the approach.

## 1 Introduction

Interaction is the key distinguishing feature of multiagent systems. We investigate the science of interaction as it underlies the engineering of multiagent systems whose constituent agents are *heterogeneous* (independently designed) and *autonomous* (independently motivated). In such systems, the internal workings of the agents take backstage to the interactions among them.

We begin from a simple yet profound question: *How may we treat interactions as first-class citizens in modeling and analyzing multiagent systems?* The usual objectives of engineering—modularly specifying, developing, composing, verifying, and validating parts—apply for interactions just as for traditional software approaches. However, existing solutions, which are designed for components such as objects, do not readily lift to interactions: an interaction somehow must simultaneously accommodate more than one perspective. Thus, importantly, the novelty of the interactive setting yields fresh and crucial technical challenges, which offer a great opportunity for multiagent systems research.

Of the many applications of multiagent systems, those in *cross-organizational* business processes provide the happy mix of practical value, theoretical subtlety, and opportunity (in the form of interest in industry) that our research community needs to sustain this research effort. Cross-organizational processes fundamentally differ from conventional software in that they are naturally modeled via interactions among *heterogeneous* and *autonomous* agents [1]. The interactions of interest are of an arms-length nature, and thus naturally understood as *communications*. In our study, we assume the existence of suitable approaches for the transmittal of information and therefore concentrate on communication understood at the level of meaning.

To engineer a multiagent system based on interactive principles presupposes a notion of the correctness of interactions among agents—in particular, here, of communications. Given such a notion, we ask if an agent is *compliant* with its expected behavior. Further, we can ask if the given agents are *interoperable* meaning that they are able to work together as expected. We can ask the above questions from the perspective of the system as a whole or of any of the participants. To formalize interoperability and compliance in interactive terms requires that we develop a theory of types using which we might modularize communications into *protocols*. We might then create repositories of protocols; determine if one protocol refines another or aggregates two or more protocols; modularly validate the protocols; modularly verify agents with respect to each relevant protocol; and so on. Notice that interfaces in object-oriented computing correspond to protocols and support constructs such as refinement and aggregation as well as the usual forms of type inference.

## 1.1 Approach

The meaning of an interaction lies at the crux of the question of its correctness. When we think at levels above the transmission of information, the meaning of communication is grounded in the relationships among the parties involved. Communication then is based on conventions by which such relationships are created, progressed (or otherwise altered), and ended. We concentrate on the contractual relationships expressed through the notion of commitments. A *commitment* involves a debtor, a creditor, an antecedent, and a consequent; it is represented as $\mathsf{C}(debtor, creditor, antecedent, consequent)$. Roughly, the debtor stakes a claim or makes a promise to the creditor about the specified consequent provided that the antecedent holds. Commitments naturally express the *what* of business relationships, and minimally constrain the *how*. For example, a commitment to pay for goods received may be discharged by paying directly, or delegated to someone who would discharge or delegate it, and so on (for any finite sequence of delegations).

In our approach, a *protocol* specifies business interactions primarily by stating how messages affect the participants' commitments. For example, returning purchased goods unopened may release the buyer from a commitment to pay. Thus many possible enactments may result from the same protocol. This is how commitments yield both rigor and flexibility. Because of its naturalness, the commitment-based approach has attracted the attention of finance and health care industry groups [2].

*Protocols* are interfaces: they constrain how agents interact, not how they are implemented. Protocols are *doubly* modular: in terms both of functionality and autonomy. For example, for functionality, an ORDER protocol between a customer and a merchant would specify only interactions dealing with order placement, leaving other functionalities to separate protocols, e.g., one for INVENTORY FULFILLMENT. Our approach enables composing protocols to yield more complex protocols, of enhanced functionality. Further, for autonomy, ORDER would specify the interactions, leaving to each the autonomous decision making of whether and how to interact, which could depend on its goals [3]. We define a *process* as the aggregation of the behaviors of the parties involved, including both their interactions and their local reasoning.

To model a process, we identify the protocols using which the different participants interact [1]. For example, a merchant and a customer may interact with each other using a NEGOTIATION protocol; the merchant, customer, and payment agency may interact via an ESCROW protocol; and, the merchant, customer, and shipper may interact through some specialized LOGISTICS protocol. When each participant acts according to its local reasoning but respecting the stated protocols, they jointly enact a multiparty business process. The contractually significant parts of the process would have been encoded in the commitments specified in the protocols; the other parts may feature only in the local policies of the participants and need not be visible externally. An agent's policies could be geared to optimize its outcomes. For example, policies would help decide what item to order, what price to quote, and so on.

The above approach obviates defining a monolithic global flow that specifies the actions of each party. Each protocol could be refined to capture additional requirements, e.g., adding receipts or guarantees to SHIPPING or PAYMENT to produce new refined protocols. Protocols can involve more than two parties; in typical usage, one partner would play multiple roles in multiple protocols [4]. For example, a purchase process may be defined as a composition of ORDER, SHIPPING, and PAYMENT protocols where the buyer in ORDER is the receiver in SHIPPING and the payer in PAYMENT.

*The potential benefits* of our protocol-based approach over traditional approaches include the following. One, for process *design*, protocols are naturally reusable whereas complete processes are not. More importantly, protocols lend themselves to modeling abstractions such as refinement and aggregation. Two, for process *implementation*, implementations of agents playing multiple roles can be more readily assembled from specifications of the roles. Three, for process *enactment*, flexible protocols enable each agent to exercise discretion via its policies or preferences even as it follows a protocol. For example, a merchant may accept only cash for discounted goods and a customer may prefer to pay for goods early or late depending upon private considerations such as of fiscal year accounting. This flexibility also enables us to capture and handle business exceptions and opportunities in a natural manner at the level of protocols. Four, for process *monitoring*, protocols provide a clean basis for determining that the interacting agents are complying with their roles in the given protocols.

## 1.2 Contributions

We motivate and characterize the key properties that would enable engineering multiagent systems with a special emphasis on applications such as cross-organizational processes. Compared to traditional formal approaches, the emphases on communications and commitments give us a novel start. By assigning meaning to communications in terms of commitments, we accomplish the following. One, we reconstruct the correctness of behaviors by characterizing *compliance* as the eventual discharge of commitments. Two, we characterize the *interoperability* of agents as the alignment of their commitments, meaning that a creditor's expectations about a commitment are met by the debtor. Three, we expand the treatment of design artifacts such as protocols by viewing them as communication types and showing how to refine and aggregate them. Using the above, we characterize the *conformance* of an agent with a role in a protocol. Further, we characterize important properties of a protocol such as its *transparency* in terms of the ability of the parties involved to verify each other's compliance. By contrast, traditional approaches (formal or otherwise) are largely confined to details such as message ordering and occurrence, and thus miss the forest for the trees.

Importantly, unlike most other multiagent systems work, our approach is undergirded by key ideas of distributed computing, especially dealing with the fact that key information is not immediately shared by all parties (even if they wish to share it). In fact, this is why protocols are important beyond plain commitments. This paper characterizes the above concepts under realistic assumptions, including multiparty settings with asynchronous communication (which aren't accommodated even in fairly recent interoperability research, e.g., [5–7]). Hence, this paper reflects crucial basic research not being addressed elsewhere. Its relevance to declarative agent languages and techniques arises from the fact that declarative representations for interaction are central to engineering robust, flexible multiagent systems, and this paper introduces and illustrates correctness criteria based on such declarative representations.

We do not introduce a formal framework in which to characterize the properties; nonetheless, we discuss the properties with rigor appropriate to illuminate their essential nature. This is consistent with our aim of motivating the properties and pointing out the challenges in their verification.

The rest of this paper is organized as follows. Section 2 introduces commitments and protocols in greater detail. Section 3 characterizes the correctness properties for interactions. Section 4 describes our contributions in relation to the most relevant literature. Section 5 lays out an ambitious agenda for multiagent systems research.

## 2 Background on Protocols and Commitments

In classical software engineering methodologies, information modeling involves the application of key abstractions such as classification, aggregation, and association among components. It would be valuable to develop similar abstractions for interactions. Notice that traditional flow-based process models don't readily support such abstractions. One, existing work imposes severely limiting assumptions to support such abstractions—refinement is specified for Petri nets restricted to one input and one output place [8], which are not as expressive as general Petri nets needed to express real processes. Two, absent a business-level semantics, the models are rigid and any deviation would be potentially erroneous, thus making it difficult to refine or generalize processes.

By contrast, protocols focus on interactions, not on implementations. Our commitment-based semantics of protocols enables us to determine if a protocol refines another protocol, and how protocols may be aggregated into other protocols. Further, we specify a protocol primarily in terms of the vocabulary for communication that it defines and only secondarily in terms of (generally, ad hoc) constraints on the ordering and occurrence of messages. By basing correctness on the discharge of commitments, we enable agents to behave flexibly. For example, a merchant may ship before receiving payment if it wishes; a customer may pay directly or via a third party; and so on. On occasion, an application may impose an otherwise ad hoc constraint. For example, in a (sit-down) restaurant, the protocol is to pay after food has been received and consumed; in a drive-through, payment precedes delivery. Such constraints often are merely guidelines for the participants and have no bearing on correctness unless they are enshrined in commitments. For example, a restaurant patron may pay early; a drive-through clerk may hand over the food before taking payment from the customer.

Flexible enactment and modeling in terms of refinement and aggregation are possible only because our semantics establishes the correctness criteria by which legitimate enactments, refinements, and aggregations can be identified [4]. Commitments express how contractual relationships form and progress during the agents' interactions. The commitment-based semantics is readily grounded via operational or messaging-level constraints [9].

*Commitments.* Contracts are key to flexible interoperation. Hohfeld [10] clarified a legal notion of contracts. Commitments cover the relevant aspects of Hohfeld's notions [11], and thus naturally represent the contractual relationships of interest.

Two main forms of commitments arise [12]: *practical* commitments are about bringing about a future condition (i.e., oriented toward tasks), whereas *dialectical* commitments [13] are about staking a claim (as in argumentation) about the past, present, or future (i.e., oriented toward assertions). The distinction between them is significant even when directed to the future. For example, I might commit dialectically that the postman will ring twice, without committing practically to ensure that the postman rings twice. This paper deals with practical commitments. For example, the customer's agreement to pay the price for the book after it is delivered is a practical commitment that the customer (as debtor) has towards the bookstore (as creditor) to ensure the price is paid.

Using commitments enables us to model interactions *computation independently* (using this term as in Model-Driven Architecture (MDA) [14]). On the one hand, commitments describe the evolving state of the ongoing business interaction and how it evolves due to the participants' communications. On the other hand, commitments help express the expectations that participants have of one another: this is the fundamental purpose of a protocol. Jointly, these enable us to readily detect and accommodate business exceptions and opportunities. Consequently, commitments lend coherence to interactions [15].

Commitments can be manipulated through a small set of operations, including create, discharge, cancel, release, assign, and delegate [11], which we lack the space to discuss here. With additional assumptions, commitments can be enforced—by penalizing agents who do not comply with their commitments.

*Protocols and commitments.* An advantage of incorporating commitments in our models is that they directly represent contractual relationships, are flexible, and lend coherence to the interactions of the participants in a process. The formalization of the specialization and generalization hierarchy of protocols is made the more interesting and useful because of the presence of commitments and roles in our model. Instead of considering uninterpreted runs (of actions and states), we consider how the commitments of the various roles evolve over different runs. The use of commitments enables more sophisticated reasoning about meaning than in traditional approaches. In particular, it enables us to characterize the similarity of states and protocol refinement in potentially subtle ways. An example is when a participant from its local perspective considers two states as interchangeable simply because it features as the creditor and debtor in the same commitments regardless of the other parties. For instance, in some settings, Alice may care only of her total accounts receivable, and not care if it is Bob or Charlie who is committed to paying her the money. In other words, instead of merely considering raw computations, it makes sense to "normalize" them in terms of commitments so as to make more precise judgments about how protocols relate to one another.

**Table 1.** A purchase protocol (customer is $c$ and merchant is $m$)

| |
|---|
| $\mathit{Offer}(m, c, \mathit{payment}, \mathit{book})$ means $\mathit{Create}(m, c, \mathit{payment}, \mathit{book})$ |
| $\mathit{Accept}(c, m, \mathit{payment}, \mathit{book})$ means $\mathit{Create}(c, m, \mathit{book}, \mathit{payment})$ |
| $\mathit{Reject}(c, m, \mathit{payment}, \mathit{book})$ means $\mathit{Release}(m, c, \mathit{payment}, \mathit{book})$ |
| $\mathit{Deliver}(m, c, \mathit{book})$ means $\mathit{Inform}(m, c, \mathit{book})$ |
| $\mathit{Pay}(c, m, \mathit{payment})$ means $\mathit{Inform}(c, m, \mathit{payment})$ |

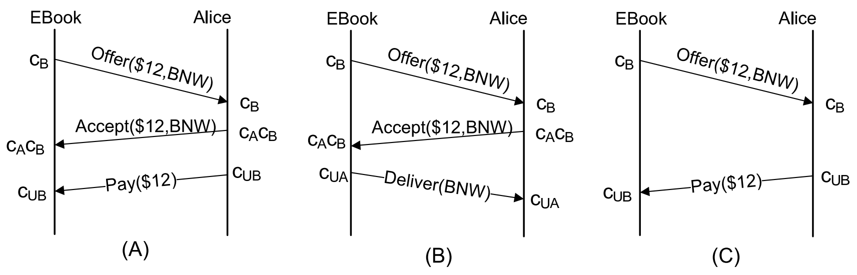

**Fig. 1.** Three *possible* enactments of protocol in Table 1

Table 1 shows the messages in a purchase protocol and their meanings. Offer from $m$ to $c$ creates $\mathsf{C}(m, c, \mathit{payment}, \mathit{book})$; Accept by $c$ creates the countercommitment $\mathsf{C}(c, m, \mathit{book}, \mathit{payment})$; $c$'s Reject releases $m$ from his commitment. Deliver means that $m$ is informing $c$ that the book has been delivered; essentially, it causes the proposition $\mathit{book}$ to hold. Pay means that $c$ is informing $m$ that the payment has been made; essentially, it causes the proposition $\mathit{payment}$ to hold. The meanings of the messages are crucial, because they help characterize the protocol declaratively. The meanings are systematically formalized in a declarative action language. Our language and technique are introduced in [16–18].

Figure 1 shows some possible enactments of the purchase protocol between a customer Alice and a merchant EBook concerning the book $\mathit{BNW}$ (for *Brave New World*) and a payment of \$12. In the figure, $c_A$ is $\mathsf{C}(\mathit{Alice}, \mathit{EBook}, \mathit{BNW}, \$12)$; $c_B$ is $\mathsf{C}(\mathit{EBook}, \mathit{Alice}, \$12, \mathit{BNW})$; $c_{UA}$ and $c_{UB}$ are the unconditional commitments $\mathsf{C}(\mathit{Alice}, \mathit{EBook}, \top, \$12)$ and $\mathsf{C}(\mathit{EBook}, \mathit{Alice}, \top, \mathit{BNW})$, respectively.

Traditional approaches force a tradeoff: checking compliance is simple with rigid automaton-based representations and difficult with flexible unconstrained reasoning agents. Commitments help us find the happy middle: protocols maximize flexibility by constraining the participants' interactions at the business level, yet provide a crisp notion of compliance: a party complies if its commitments are discharged, no matter if delegated or otherwise manipulated.

*Protocols and computations.* In essence, each protocol allows a set of computations or runs, each run being an alternative that requires a specific sequence of actions upon the participants. Two basic intuitions about protocol refinement are that (1) a more general protocol includes additional runs (more ways to satisfy) beyond those in a less general protocol; and (2) a more general protocol includes shorter runs (fewer steps to satisfy) than a less general protocol.

Our commitment-based semantics yields a rigorous basis for protocol *refinement* and *aggregation* [19]. In principle, these properties enable reusing protocols from a repository. For example, PAYMENT BY CHECK refines PAYMENT. Further, ORDER, PAYMENT, and SHIPPING can be combined into a new protocol for PURCHASE. This composed protocol would capture the reusable interactions and service agreements that underlie a business process. For example, PURCHASE would specify how orders may be placed, payments made, and shipping arranged. When protocols are composed, so are the roles; e.g., the payer in PAYMENT may be composed with the receiver in SHIPPING. Multiple copies of the same protocol may be composed: in an ARBITRAGE protocol, the arbitrageur role would compose the seller role in one copy of PAYMENT with the buyer role in the second copy.

As in other formal semantics, the runs are merely abstract entities used to establish logical properties. We would never explicitly enumerate the potentially infinite number of possible runs, but we can use the abstract definition to show important algebraic relationships. Mallya & Singh [19] show important progress, but their approach is far from complete. Specifically, it deals with sets of runs, but does not apply directly on protocol specifications as one would find in a repository.

## 3 Correctness Properties

We begin by motivating some key definitions. Notice that although the above discussion uses protocols as design artifacts, compliance and interoperability apply without regard to any protocol. Although our main definitions and methods are oriented toward commitments, they are undergirded by considerations of distributed computing, especially of asynchrony in messaging.

### 3.1 Interoperability

The *interoperability* of a set of roles or agents, regardless of protocol, means that they jointly meet the expectations they place on each other. Some aspects of interoperability depend on meanings; others on the messaging system that underlies communications.

We assume that messaging is asynchronous, reliable, and pairwise (for each sender and receiver) order-preserving: this matches what emerging middleware standards [20] offer. Thus in two-party cases, each party would eventually learn of the relevant moves and expectations of the other: the only kind of pathology possible is that the parties may view some pairs of messages in opposite orders. In multiparty cases, the messaging conditions can become more subtle: e.g., a party would lack direct information about messages exchanged among other parties. Mostly, this is a good thing because the parties can proceed with minimal mutual dependencies. However, when such information materially affects a desired property, we would need to change either the requirements (so information about remote events becomes irrelevant) or the specification (so that the necessary information flows to the appropriate parties).

Interoperation classically is treated as a conjunction of liveness and safety. To these we add alignment.

**Liveness** means that progress will take place: desirable states will be visited infinitely often. Liveness can fail if a receiver blocks (awaiting a message that is never sent). For example, let Buyer-A demand delivery before payment and Seller-A demand payment before delivery. Now, Buyer-A and Seller-A would deadlock, each awaiting the other's message.

**Safety** means that the system doesn't enter an undesirable state: agents must be ready to receive the messages being sent to them. Safety is best understood in a multiparty setting. If a buyer expects to receive a confirmation before a shipment but receives them in the opposite order, its resultant state is not defined. We should ensure the messages occur in only those orders that the buyer accepts.

We apply causality [21] to model the above concepts. The sending of a message is causally prior to its receipt; for any two locally ordered events (sends or receives), the first is (potentially) causally prior to the second: "potential" because from external observations we cannot infer if the two events are truly related. We can infer true causality from the agents' specifications, in settings where the specifications are available. We can characterize liveness and safety in terms of the compatibility among causal orders involving receives and sends. We conjecture that the above will yield superior solutions to those in the recent distributed computing literature, e.g., [5–7]. The literature considers two-party cases or violates substitutability: that substituting an agent with a conforming agent must preserve interoperability.

**Alignment** is interoperability with respect to expectations at the level of meaning: do the participants agree about the states of their commitments to each other? A set of agents or roles is *aligned* provided throughout any enactment, whenever one concludes it is the creditor of a commitment, the corresponding debtor $x$ concludes that $x$ is the debtor of the commitment [22]. In other words, the debtor recognizes a commitment that the creditor expects of it. How commitments are created, discharged, and manipulated depends on the messages sent and received.

From the point of view of interoperability, interesting agent specifications are of two kinds: constitutive and regulative [22]. An agent's constitutive specification deals only with the meaning of messages. In other words, it specifies what messages *count as* for the agent. An agent's regulative specification, in contrast, describes agent behavior; i.e., it describes the conditions under which the agent sends and receives particular messages. Regulative specifications are thus closer to implementations.

Agents could be misaligned if, in their constitutive specifications, messages are interpreted differently. For example, if the buyer and seller interpret the Offer message as different commitments, they would be misaligned [22] even though they satisfy safety.

Judging the constitutive alignment of a set of agents by statically analyzing their specifications is nontrivial because message meanings may be conditional, and thus potentially affected by how other messages change the relevant conditions. For example, if one message constitutes an authorization and the meaning of a second message relies upon that authorization, the commitments resulting from the second message would depend upon whether the first message precedes it.

Agents could also become misaligned due to asynchrony: the debtor's and the creditor's conclusions about a commitment may conflict because they see different messages occurrences or orders. Delegations and assignments of commitments inherently involve three parties, and are thus even more challenging.

A specification may fail safety or liveness without failing alignment. We saw above that Buyer-A and Seller-A fail liveness. However, they may never disagree about their commitments and hence would satisfy alignment.

## 3.2 Conformance and Operability

Conformance and operability apply to each interoperability property: liveness, safety, and alignment. A role *conforms* to, i.e., is a subtype of, another role provided the first role meets all expectations placed on the second and holds no expectations of others beyond what the second does. Similarly, an agent conforms to, i.e., instantiates, a role. Conformance is important because it helps us build a library of roles without which engineering would lapse into one-off solutions. To handle conformance properly would require considering the semantics of protocols not in terms of simple runs, but in terms of the choices they afford each role. Echoing the intuition of alternating refinement [23], expectations placed on a role correspond to "external" choices; expectations held by a role correspond to "internal" choices.

A protocol is *operable*, i.e., potentially enactable, if the roles it specifies are interoperable. A protocol may fail to be operable when it requires a role to act based on events that the role cannot observe. Operability is an important quality criterion for protocols: ideally, the protocols in a library should be operable, so developers may implement selected roles conformantly, and be assured of interoperation.

Let protocol FLEXIBLE PURCHASE allow a payment to occur before or after the delivery of goods. It is easy to see that Buyer-A and Seller-A (introduced above), respectively, conform to the customer and merchant roles in FLEXIBLE PURCHASE. Recall, however, that Buyer-A and Seller-A together fail liveness. Hence FLEXIBLE PURCHASE is not operable for liveness. Conversely, let PREPAID PURCHASE require payment to occur before delivery. Then, any pair of conforming customer and merchant would be live and safe. Hence, PREPAID PURCHASE is operable. Buyer-A is nonconformant with the customer role, whereas Seller-A is conformant with the merchant role of PREPAID PURCHASE. Seller-A and Buyer-A failing liveness doesn't mean PREPAID PURCHASE is inoperable: it is Buyer-A that is messed up.

## 3.3 Compliance and Transparency

*Compliance* means that each agent performs as expected by others, by discharging its commitments. We can prove compliance only when we know each agent's specification and relevant assumptions about the environment hold. That is, compliance can be verified for specific runs but not proved in general for open systems [24]. Notice that alignment and compliance are independent of each other: e.g., an interoperable buyer may be committed to pay, but may refuse to do so. An agent may *verify* a debtor's compliance based on its observations in a specific enactment. Assuming that the discharge of a commitment is observable (e.g., occurs via a message), verifying compliance is simple in two-party cases. If a debtor complies, the creditor would eventually know. If a debtor does not comply, then the creditor would eventually know—provided the commitment includes a deadline. In multiparty cases, a creditor may lack some important observations, and hence special techniques would be required to verify alignment.

A protocol is *transparent* if each role in it can verify the compliance of its debtors. However, not all protocols enable each role to verify compliance at runtime: a protocol may be such that "news" relevant to a commitment might not be propagated to the creditor. Transparency is an important quality criterion for protocols: it ensures that participants can verify if others are not complying.

### 3.4 Refinement and Compatibility

The *refinement* of a protocol by another protocol means that the second protocol generates only computations that are allowed by the first. Modeling via commitments enables us to finesse the intuitions about protocol refinement. For example, a simple PAYMENT protocol might require that the payer transfer funds to the payee. A particular refinement of this might be PAYMENT WITH A CHECK. To pay with a check, the payer would send a check to the payee, who would deposit the check at his bank, which would present it to the payer's bank, which would send the funds to the payee's bank, which would make those funds available to the payee. Thus PAYMENT BY CHECK is a specialization of PAYMENT, but it involves additional roles and steps, and skips some of the steps of PAYMENT, e.g., direct transfer. With a commitment-based definition, we can formally establish that PAYMENT BY CHECK refines PAYMENT—something that would not be possible with traditional approaches because of the above differences between the two protocols. The key intuition is that the commitments at critical states line up correctly. This is a significant departure from traditional notions of refinement which, because they lack commitments, insist upon the computations to match in their detailed steps.

Notice that an agent designed to play a role in a refined protocol may not comply with any role in the original protocol. This is because the agent may not interpret messages in a way compatible with the original protocol. For example, in PAYMENT BY CHECK, a merchant may interpret a check as being adequate as a receipt (once it is cleared and returned to the customer by the customer's bank), but the customer may not interpret it like that and may continue to expect a separate receipt as in PAYMENT. Further, the agent may fail to interoperate with roles defined in the original protocol. This is because it may send messages that are not defined in the original protocol. In general we would not be able to substitute a role from a refined protocol for a role in the original protocol. The foregoing is motivation for the property of *compatibility*, which determines if roles in one protocol conform to roles in another protocol.

Table 2 summarizes the above properties. With the exception of compliance, these properties can be verified by a static analysis of the appropriate specifications.

**Table 2.** The properties summarized

| **Property** | **Of What?** |
|---|---|
| Refinement, compatibility, operability, transparency | Protocols |
| Interoperability (safety, liveness, or alignment) | Agents and roles |
| Conformance | Roles |
| Compliance | Agents |

## 4 Discussion: Relevant Literature

Our main contribution in this paper is in characterizing the key correctness properties that would support an interaction-oriented approach to building software systems, particularly cross-organizational business processes. In particular, the correctness properties reflect high-level requirements of such systems.

Interestingly, Parnas [25] proposed early in the study of software architectures that connectors be treated not as control or data flow constructs but as *assumptions* made by each component about the others. Arguably, much of the subsequent work on software architecture regressed from Parnas' insight: it has primarily considered connectors at the level of flow, e.g., dealing exclusively with message order and occurrence [26]. In formulating the assumptions at a high level, we see a great opportunity for multiagent systems research to address some of the long-standing challenges in software.

*Conventional formal methods.* Current modeling formalisms, such as finite state machines and Petri Nets, originated in distributed computing and apply at lower levels of abstraction than needed for flexible business interactions [27, 8]. When applied to business protocols, these formalisms result in specifications that are over-constrained to the level of specific sequences of actions. Recent approaches have sought to express scheduling requirements declaratively, via temporal logic [28–30]. Although they are more flexible and modular than operational representations, these approaches do not express business semantics.

FIPA, the Foundation for Intelligent and Physical Agents (now part of IEEE) recognized the importance of reusable interaction protocols in the late 1990s [31]. Odell *et al.* [32] give one of the earliest uses of UML for protocols. They show how various UML diagrams can be applied for modeling agent interactions. This work shows about how far you can go in a conventional software framework, and has inspired our work. The present paper is about fundamental enhancements to conventional models to capture protocols and their commitment-based semantics.

Leading approaches model conversations via finite-state machines and establish properties such as how roles may realize a protocol or a protocol subsumes another [33, 34]. Dastani *et al.* [35] show how to model a rich family of coordination connectors for multiagent systems. Honda *et al.* [36] develop a type theory that would support multiparty sessions: in essence this would help robustly generate roles. These works formalize protocols as data and control flow abstractions. They do not consider the meaning of messages and thus lack the business-level semantics that distinguishes our work. However, their treatment of messages and computations at a low level is useful, and complementary to our work.

Whereas deontic logic only deals with what is obligatory or permissible and thus disregards an agent's obligations *to* another agent, commitments are directed and context sensitive. Commitments include support for a variety of operations [11, 37]. Foster *et al.* [38] seek to capture the semantics of process interactions via the notion of obligation policies. Obligations are rather weak in their formulation, however. Specifically, obligations are not reified, and cannot be manipulated to capture flexible interactions among independent parties. Lomuscio *et al.* [39] formalize correctness properties in a temporal logic and show how to verify them. They consider obligations but do not consider commitments as here. Lomuscio *et al.* also concentrate on only one correctness property, which is somewhat like compliance.

*Business processes.* The MIT Process Handbook (MITPH) [40] is of great relevance intellectually. MITPH includes an extensive classification and systematic organization of business processes based on two dimensions of process hierarchies, one that composes the *uses* of a process out of its constituent *parts*, and another that subclasses *generalizations* of a process into *specializations*. Our work can provide the rigorous underpinnings for work such as the MITPH. Grosof and Poon [41] develop a system to represent and execute business rules from MITPH. Wyner and Lee [42] study specialization for data flow diagrams. Their approach can form the basis of the processes identified in MITPH. These concepts turn out to be complex and not readily applied to entire business processes. Further, since Wyner and Lee do not capture the content through a high-level representation such as commitments, the results are not intuitive.

Our approach agrees with the newer declarative forms of artifacts-based process modeling [43] in terms of deemphasizing low-level operational details in favor of business semantics. However, these approaches do not have a central organizing principle on par with commitments, and thus do not offer a generic and flexible basis for determining the properties we introduced above.

*Agent communications.* Fornara and Colombetti [44] describe how commitments relate to FIPA messages, demonstrating this with an example. Rovatsos [45] proposes a commitment-based semantics for communications under synchronous messaging. His approach violates autonomy by legislating agent behaviors from within the language specification: this level of prescription is ill-suited to most multiagent applications.

Yolum and Singh [46] [47] offer one of the first accounts of the use of commitments in modeling protocols to improve flexibility for participating agents, which was enhanced by Winikoff *et al.* [48]. Johnson *et al.* [49] develop a scheme for identifying when two commitment-based protocols are equivalent. Their scheme, however, is simplistic, classifying protocols based solely on their syntactic structure. Our work provides stronger results from an application point of view and relates better to Web services.

Commitments have found application in formalizing argumentation, e.g., [50, 51]. Usually, though, this work makes simplifying assumptions such as (1) maintaining a unique commitment store; (2) informally specifying the meanings of communicative acts as effects on the store; (3) assuming synchronous two-party communications.

*Agent-oriented software engineering (AOSE).* A number of useful software methodologies for building multiagent systems for IT applications have emerged that incorporate rich metamodels and describe how to build a series of software artifacts [52, 53, 3]. Garcia-Ojeda *et al.* [54] synthesize existing metamodels into a comprehensive metamodel of organizations geared toward process modeling. We recently developed Amoeba, a protocol-based methodology compatible with the ideas of this paper [1].

The above methodologies address the challenges of autonomy and heterogeneity by giving prominence to communication. Such works are clearly valuable and worthwhile. However, current approaches do not consider the full subtleties both of meaning and of distribution. By contrast, this paper addresses the foundations for business interactions understood in terms of commitments. The proposed definitions will offer a foundations for building a new family of tools that, in principle, could be used within any of the above methodologies, because they all support aspects of interaction and of agents playing roles in interactions.

## 5 Conclusions and Directions

This paper presents a key step in our program of research to develop underpinnings of multiagent systems—and indeed, of all software—on interactive grounds with an emphasis on declarative formulations. The main point to take away is the richness of the correctness properties. These properties echo well-known conventional properties but their characterization in a declarative, interactive setting adds a lot of subtlety that traditional approaches cannot express. The foregoing leads to two broad questions.

– *Theory*. What are practical decision algorithms for these properties? How can we specify agents who may play specified roles (while applying their local policies)? How can we determine that agents (supposedly) enacting a protocol are complying with the protocol? What are practical algorithms for judging the varieties of interoperability, conformance, operability, compliance, and transparency?
– *Suitability and applicability*. Does representing meaning via commitments provide a sufficiently natural basis for business interoperation? How readily can meaning be associated with tools to engineer and use protocols? Can we specify commitments sufficiently precisely in real-life business settings? How can we use the above properties and algorithms to enable protocol design and agent implementation?

The above questions constitute a substantial research agenda. Addressing this agenda presupposes an adequate formalization of commitments. Recent work on the formal semantics of commitments [12] and commitment operations [9] are steps in that direction.

## References

1. Desai, N., Chopra, A.K., Singh, M.P.: Amoeba: A methodology for modeling and evolution of cross-organizational business processes. ACM Transactions on Software Engineering and Methodology (TOSEM) **19**(2) (April 2010) To appear.

2. Desai, N., Chopra, A.K., Arrott, M., Specht, B., Singh, M.P.: Engineering foreign exchange processes via commitment protocols. In: Proceedings of the 4th IEEE International Conference on Services Computing (SCC), Los Alamitos, IEEE Computer Society Press (2007) 514–521
3. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Journal of Autonomous Agents and Multi-Agent Systems **8**(3) (May 2004) 203–236
4. Desai, N., Mallya, A.U., Chopra, A.K., Singh, M.P.: Interaction protocols as design abstractions for business processes. IEEE Transactions on Software Engineering **31**(12) (December 2005) 1015–1027
5. Baldoni, M., Baroglio, C., Martelli, A., Patti, V.: A priori conformance verification for guaranteeing interoperability in open environments. In: Proceedings of the 4th International Conference on Service-Oriented Computing (ICSOC). Volume 4294 of LNCS., Springer (2006) 339–351
6. Fournet, C., Hoare, C.A.R., Rajamani, S.K., Rehof, J.: Stuck-free conformance. In: Proceedings of the 16th International Conference on Computer Aided Verification (CAV). Volume 3114 of LNCS., Springer (2004) 242–254
7. Giordano, L., Martelli, A.: Verifying agent conformance with protocols specified in a temporal action logic. In: Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence (AI*IA). Volume 4733 of LNCS., Springer (2007) 145–156
8. van der Aalst, W., van Hee, K.: Workflow Management Models, Methods, and Systems. MIT Press, Cambridge, MA (2002)
9. Chopra, A.K., Singh, M.P.: Multiagent commitment alignment. In: Proceedings of the 8th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS), Columbia, SC, IFAAMAS (May 2009) 937–944
10. Hohfeld, W.N.: Fundamental Legal Conceptions as Applied in Judicial Reasoning and other Legal Essays. Yale University Press, New Haven, CT (1919) A 1919 printing of articles from 1913.
11. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law **7** (1999) 97–113
12. Singh, M.P.: Semantical considerations on dialectical and practical commitments. In: Proceedings of the 23rd Conference on Artificial Intelligence (AAAI), Menlo Park, AAAI Press (July 2008) 176–181
13. Walton, D.N., Krabbe, E.C.W.: Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning. State University of New York Press, Albany (1995)
14. OMG: The Object Management Group's Model Driven Architecture (MDA) (2006) http://www.omg.org/mda/.
15. Jain, A.K., Aparicio IV, M., Singh, M.P.: Agents for process coherence in virtual enterprises. Communications of the ACM **42**(3) (March 1999) 62–69
16. Chopra, A.K., Singh, M.P.: Contextualizing commitment protocols. In: Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems. (2006) 1345–1352
17. Desai, N., Singh, M.P.: A modular action description language for protocol composition. In: Proceedings of the 22nd Conference on Artificial Intelligence (AAAI), Menlo Park, AAAI Press (July 2007) 962–967
18. Desai, N., Singh, M.P.: On the enactability of business protocols. In: Proceedings of the 23rd Conference on Artificial Intelligence (AAAI), Menlo Park, AAAI Press (July 2008) 1126–1131
19. Mallya, A.U., Singh, M.P.: An algebra for commitment protocols. Journal of Autonomous Agents and Multi-Agent Systems **14**(2) (April 2007) 143–163

20. AMQP: Advanced message queuing protocol (2007) http://www.amqp.org.
21. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communications of the ACM **21**(7) (July 1978) 558–565
22. Chopra, A.K., Singh, M.P.: Constitutive interoperability. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS). (May 2008) 797–804
23. Alur, R., Henzinger, T.A., Kupferman, O., Vardi, M.Y.: Alternating refinement relations. In: Proceedings of the 9th International Conference on Concurrency Theory (CONCUR). Volume 1466 of LNCS., Springer (1998) 163–178
24. Venkatraman, M., Singh, M.P.: Verifying compliance with commitment protocols: Enabling open Web-based multiagent systems. Autonomous Agents and Multi-Agent Systems **2**(3) (September 1999) 217–236
25. Parnas, D.L.: Information distribution aspects of design methodology. In: Proceedings of the International Federation for Information Processing Congress. Volume TA-3., Amsterdam, North Holland (1971) 26–30
26. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Signature Series. Addison-Wesley, Boston (2004)
27. Harel, D., Gery, E.: Executable object modeling with statecharts. IEEE Computer **30**(7) (July 1997) 31–42
28. Singh, M.P.: Distributed enactment of multiagent workflows: Temporal logic for service composition. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS), New York, ACM Press (July 2003) 907–914
29. Pesic, M., Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P.: Constraint-based workflow models: Change made easy. In: Proceedings of the On the Move to Meaningful Internet Systems (Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS), Part I. Volume 4803 of LNCS., Springer (2007) 77–94
30. Wu, Q., Pu, C., Sahai, A., Barga, R.S.: Categorization and optimization of synchronization dependencies in business processes. In: Proceedings of the 23nd International Conference on Data Engineering (ICDE), IEEE (2007) 306–315
31. FIPA: FIPA interaction protocol specifications (2003) FIPA: The Foundation for Intelligent Physical Agents, http://www.fipa.org/repository/ips.html.
32. Odell, J., Parunak, H.V.D., Bauer, B.: Representing agent interaction protocols in UML. In: Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE). (2001)
33. Benatallah, B., Casati, F., Toumani, F.: Representing, analysing and managing web service protocols. Data & Knowledge Engineering **58**(3) (2006) 327–357
34. Bultan, T., Fu, X., Hull, R., Su, J.: Conversation specification: A new approach to design and analysis of e-service composition. In: Proceedings of the Twelfth International World Wide Web Conference (WWW). (2003) 403–410
35. Dastani, M., Arbab, F., de Boer, F.S.: Coordination and composition in multi-agent systems. In: Proceedings of the 4rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), ACM (2005) 439–446
36. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. In: Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), ACM (2008)
37. Singh, M.P., Chopra, A.K., Desai, N.: Commitment-based SOA. IEEE Computer **42** (2009) Accepted; available from http://www.csc.ncsu.edu/faculty/mpsingh/papers/.
38. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Model-based analysis of obligations in web service choreography. In: Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW). (2006) 149–156

39. Lomuscio, A., Qu, H., Solanki, M.: Towards verifying compliance in agent-based web service compositions. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS), Columbia, SC, International Foundation for Autonomous Agents and MultiAgent Systems (2008) 265–272
40. Malone, T.W., Crowston, K., Herman, G.A., eds.: Organizing Business Knowledge: The MIT Process Handbook. MIT Press, Cambridge, MA (2003)
41. Grosof, B.N., Poon, T.C.: SweetDeal: Representing agent contracts with exceptions using XML rules, ontologies, and process descriptions. In: Proceedings of the 12th International Conference on the World Wide Web. (2003) 340–349
42. Wyner, G.M., Lee, J.: Defining specialization for process models. In: [40]. MIT Press (2003) 131–174
43. Hull, R.: Artifact-centric business process models: Brief survey of research results and challenges. In: On the Move to Meaningful Internet Systems. Volume 5332 of LNCS., Springer (2008) 1152–1163
44. Fornara, N., Colombetti, M.: Defining interaction protocols using a commitment-based agent communication language. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), New York, ACM Press (July 2003) 520–527
45. Rovatsos, M.: Dynamic semantics for agent communication languages. In: Proceedings of the 6th international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2007) 1–8
46. Yolum, P., Singh, M.P.: Commitment machines. In: Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL-01). Volume 2333 of LNAI., Springer-Verlag (2002) 235–247
47. Yolum, P., Singh, M.P.: 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 (AAMAS), New York, ACM Press (July 2002) 527–534
48. Winikoff, M., Liu, W., Harland, J.: Enhancing commitment machines. In: Proceedings of the 2nd International Workshop on Declarative Agent Languages and Technologies (DALT). Volume 3476 of LNAI., Berlin, Springer-Verlag (2005) 198–220
49. Johnson, M.W., McBurney, P., Parsons, S.: When are two protocols the same? In Huget, M.P., ed.: Communication in Multiagent Systems: Agent Communication Languages and Conversation Policies. Volume 2650 of LNAI. Springer-Verlag, Berlin (2003) 253–268
50. Amgoud, L., Maudet, N., Parsons, S.: An argumentation-based semantics for agent communication languages. In: Proceedings of the 15th European Conference on Artificial Intelligence (ECAI), IOS Press (2002) 38–42
51. Norman, T.J., Carbogim, D.V., Krabbe, E.C.W., Walton, D.N.: Argument and multi-agent systems. In Reed, C., Norman, T.J., eds.: Argumentation Machines. Kluwer (2004)
52. Bergenti, F., Gleizes, M.P., Zambonelli, F., eds.: Methodologies and Software Engineering for Agent Systems. Kluwer, Boston (2004)
53. Henderson-Sellers, B., Giorgini, P., eds.: Agent-Oriented Methodologies. Idea Group, Hershey, PA (2005)
54. Garcia-Ojeda, J.C., DeLoach, S.A., Robby, Oyenan, W.H., Valenzuela, J.: O-MaSE: A customizable approach to developing multiagent processes. In: Proceedings of the 8th International Workshop on Agent Oriented Software Engineering (AOSE 2007). (2007)