On the Enactability of Business Protocols

Nirmit Desai
IBM India Research Lab
Embassy Golf Links, Block D
Bangalore 560071, India
nirmitv@gmail.com

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

Abstract

Protocols specifying business interactions among autonomous parties enable reuse and promote interoperability. A protocol is specified from a global viewpoint, but enacted in a distributed manner by (agents playing) different roles. Each role describes a local representation. An ill-specified protocol may yield roles that fail to produce correct enactments of the protocol. Existing approaches lack a formal and comprehensive treatment of this problem. Building on recent work on declaratively specifying a protocol as a set of rules of causal logic, this paper formally defines the enactability of protocols. It presents necessary and sufficient conditions for the enactability of a protocol as well as a decision procedure for extracting correct roles from enactable protocols.

Introduction

Much of commerce today is conducted via standardized electronic business interactions on the Web. RosettaNet (www.rosettanet.org) is a widely deployed standard for supply chains. Other notable efforts include ebBP (oasis-open.org/specs/#ebxmlbp2.0.4), FIPA (fipa.org/repository/ips.html), WS-CDL (w3.org/TR/ws-cdl-10), TWIST foreign exchange processes (twiststandards.org), and the MIT Process Handbook (Malone, Crowston, & Herman 2003).

A *choreography* specifies a set of roles and their allowed interactions from a global viewpoint in terms of the ordering and occurrence of possible messages. A (business) *protocol* is a declarative specification of a choreography augmented with the meanings of the messages. The declarative specification of message meanings is an important contribution from AI that results in improved modeling and enactment of protocols (Desai et al. 2007). Thus protocols can better address the shared goal of the above standards, namely, to promote interoperation and reuse in Web processes.

To participate in a protocol, a business partner must enact one of the protocol's roles. Enacting roles is nontrivial because of the inherent distribution of Web processes. Different agents playing different roles may observe different messages and possibly in different orders. Thus, although a protocol may require or constrain an agent's action (as it plays a role) upon its observing a message, the agent may never observe that message, and thus inappropriately act or fail to act. This problem is exacerbated by asynchrony which is motivated for conceptual and practical reasons peculiar to the Web: the autonomy of the agents and the performance hit of enforcing synchrony despite high latency.

A protocol specifies the desirable interactions. A correct protocol should not rely upon any additional restrictions in order to yield correct enactments. We examine the specification of a protocol to determine if it may be enacted correctly. Whenever a protocol is specified, its enactability is a concern. This points to the need for formalizing enactability to provide enhanced tools. Indeed, Desai *et al.* (2007) identify one of the challenges addressed here (of nonlocal choice) in enacting TWIST protocols.

This paper goes beyond the state of the art in addressing the following important questions in business protocols design. What are the specific ways in which protocols may fail to be enactable? What properties of protocols preclude the above problems, and how may we check such properties? How do we extract correct roles from enactable protocols?

Technical Framework

We model protocols in the action description language $C+$, which is based on causal logic (Giunchiglia *et al.* 2004). The following concepts and $C+$ constructs are sufficient for the purposes of this paper.

- A fluent is a condition that holds or does not hold, e.g., "request for quote is received." An inertial fluent persists unless explicitly changed.
- An action may change the truth value of the fluents. Exogenous actions have causes external to the model (in our case, external to the protocol under consideration).
- A rule $\ulcorner$nonexec A if $\psi\urcorner$ prevents the occurrence of an action formula A on any transition from a state where ψ holds, where ψ can be a formula of actions and fluents.
- A rule $\ulcorner$caused A if $\psi\urcorner$ specifies that an action formula A must hold on a transition from a state in which ψ holds.
- A transition system consists of states and transitions among them. A state is an interpretation of all fluents and a transition label is an interpretation of all actions. A transition system is a model of the $C+$ specification if it respects all its causal rules.

A *protocol* and a *role* each are given by $C+$ specifications whose rules specify the meanings of and constraints on the messages to be exchanged. A protocol exhibits a global viewpoint in that its rules talk about message *exchanges* (as if send and receive were a single atomic event). A role, on the other hand, exhibits a local viewpoint in that its rules treat sending and receiving a message as separate actions, and treat the communication channels explicitly.

Let us first consider protocols. A message m from role α to role β is written $m(\alpha, \beta)$. For each message m_i, we define an exogenous action symbol m_iA to denote the action of exchanging m_i, and an inertial fluent symbol m_iF to record the fact that m_i is exchanged. Modeling the messages as exogenous actions reflects the autonomy of the agents playing a role: a protocol doesn't "explain"—in the sense of causal logic—why agents act the way they do.

The business meaning of a message is given in terms of *commitments*. A commitment $\mathsf{C}(x, y, p, q)$ means that debtor x is committed to creditor y to do consequent q if antecedent p is done. Unlike obligations, commitments are directed, contextual, and manipulable (Singh 1999). For example, a message may *create*, *delegate*, *assign*, or *cancel* a commitment. A message may also cause antecedent or a consequent of a commitment, thereby detaching or discharging it, respectively. For example, if p is caused, the above commitment is detached and becomes unconditional: $\mathsf{C}(x, y, \mathsf{T}, q)$. If q is caused the commitment is discharged.

Our protocol specification language includes the following schemas. Commitments are inertial fluents and commitment antecedents, consequents, and operations are actions.

SCHEMA$_1$. nonexec $m_1\mathrm{A} \wedge m_2\mathrm{A}$

SCHEMA$_2$. nonexec $m\mathrm{A}$ if ψ_f

SCHEMA$_3$. caused a if ψ

SCHEMA$_4$. caused $m\mathrm{F}$ if $m\mathrm{A}$

$\psi_f ::= \bot \mid \top \mid mF \mid op \mid \neg\psi_f \mid \psi_f \wedge \psi_f \mid \psi_f \vee \psi_f$
$\psi ::= mA \mid \bot \mid \top \mid mF \mid op \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi$
$op ::= \mathit{create} \mid \mathit{delegate} \mid \mathit{assign} \mid \mathit{cancel} \mid \mathit{release}$
$a ::= op \mid antecedent \mid consequent$

Note that ψ_f does not include message actions. Due to SCHEMA$_3$, the only actions that can be caused explicitly are the commitment operations, antecedents, and consequents. These actions manipulate the commitment fluents. In the rest of the paper, ψ, ψ_1, and ψ_2 are as ψ above and ψ_f is as ψ_f above. SCHEMA$_4$ is added automatically to the specification for each message in a protocol.

For example, an order placement protocol *Ord* between a buyer and a seller may be specified as follows.

ORD$_1$. caused *create*(C(S, B, pay, goods)) if quoteA

ORD$_2$. caused *create*(C(B, S, goods, pay)) if acceptA

ORD$_3$. nonexec acceptA $\wedge$ rejectA

ORD$_4$. nonexec reqForQuoteA if reqForQuoteF

ORD$_5$. nonexec quoteA if quoteF $\vee$ $\neg$reqForQuoteF

ORD$_6$. nonexec acceptA if acceptF $\vee$ $\neg$quoteF $\vee$ rejectF

ORD$_7$. nonexec rejectA if rejectF $\vee$ $\neg$quoteF $\vee$ acceptF

ORD$_1$ specifies the meaning of quote as creating a commitment from the seller to the buyer that if the buyer makes a payment, the seller would deliver the goods. ORD$_2$ specifies the meaning of accept. ORD$_3$ specifies that accept and reject are mutually exclusive. Each of the remaining rules make a message nonrepeatable by making it nonexecutable when its corresponding fluent holds. ORD$_4$ has no other purpose. ORD$_5$ prevents quote if reqForQuote has not happened. ORD$_6$ prevents accept if quote has not yet happened or reject has already happened. ORD$_7$ prevents reject similarly.

Infrastructure Assumptions. We assume that the messaging infrastructure is reliable (no messages lost, duplicated, or spuriously sent), point-to-point, asynchronous, and supports an unbounded random access inbox for each role. As in email or groupware settings, an agent playing a role may choose what messages to read and may read them out of order; reading a message removes it from the inbox. Challenges similar to those addressed in this paper would arise with FIFO channels, and can be addressed similarly.

Extracting Roles

Because progress depends on the agent's private policies, it cannot be guaranteed from the standpoint of protocols. In general, even verifying if a set of agents will progress is undecidable when they communicate via unbounded asynchronous channels (Bultan, Su, & Fu 2006). We emphasize enactability, which simply means that any progress made by agents playing roles is correct with respect to the protocol. Since we consider only protocols and roles, below we use the term *role* as shorthand for role or "any agent enacting a role."

Given a protocol, how can we extract the roles that would correctly enact a protocol? The root of the problem is that, in practical settings, message exchanges are asynchronous: the sender doesn't block for the receiver. Also, the messages are point-to-point, implying only the sender and the receiver of a message have knowledge of the message exchange. Even in a two-party protocol, the roles may observe the messages in different orders. Thus, in general, in protocols with three or more roles, the roles would be unaware of the progress being made by others.

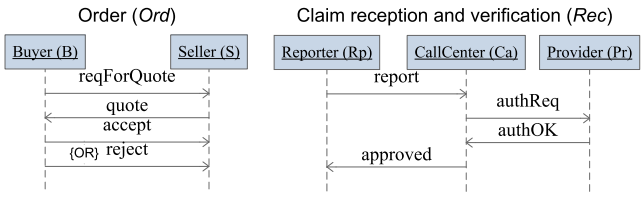

Figure 1: Typical conversations of protocols *Ord* and *Rec*

Definition 1 A *conversation* is a possible enactment of a protocol. A protocol *generates* a conversation if the messages in the conversation occur on any path consistent with a transition system that models the $C+$ specification of the protocol. A set of roles *generates* a conversation if each role sends and receives its respective messages in the conversation on any path consistent with a transition system that

models the $C+$ specification of the role.

Figure 1 shows sample conversations for *Ord* and the *Rec* protocol introduced below. The above leads to the definition of *correctness* of a set of roles.

Definition 2 A set of roles is *correct* with respect to a protocol iff the roles generate all and only the conversations generated by the protocol.

Extracting a role from a protocol amounts to choosing and transforming a relevant subset of protocol rules to ensure correct behavior for the selected role. The relevant rules are those that apply to the messages that the role sends or receives and the commitments of which the role is either the debtor or creditor. In general, the role may not be aware of some of the fluents or actions in the premises of some of the chosen rules. Such premises need to be adjusted.

Each role owns a knowledge base (KB) and an inbox. Only a KB owner can read and write to it; only senders can add, and only the owner (i.e., the addressee) can remove, messages from an inbox.

Each message action $mA(\alpha, \beta)$ in a protocol, yields a message sending action $!m$ in role α and a message receiving action $?m$ in role β. An inertial fluent $|m|$ denotes a message in an inbox, logically in transit. A message action $mA(\alpha, \beta)$ yields separate fluents mF for α and β, and a fluent $|m|$ in β's inbox. Each protocol rule ⌜caused mF if mA(α, β)⌝ maps to rules for α and β as follows:

Sender:α	Receiver:β
caused mF if !m	caused mF if ?m
caused |m| if !m	caused ¬|m| if ?m
	nonexec ?m if ¬|m|

The first rules for each role record the occurrence of the message. The second sender rule models the production of the message in the inbox of the receiver whereas the second receiver rule models the consumption of the message from the receiver's inbox. The third receiver rule stipulates that a message must exist in its inbox for it to be received. These rules show how reception causally depends on sending.

For example, a buyer (B) extracted from *Ord*:

B$_1$. nonexec !reqForQuote if reqForQuoteF

B$_2$. nonexec ?quote if ¬reqForQuoteF ∨ quoteF

B$_3$. caused *create*(C(S, B, pay, goods)) if ?quote

B$_4$. nonexec !accept if ¬quoteF ∨ acceptF ∨ rejectF

B$_5$. nonexec !reject if ¬quoteF ∨ rejectF ∨ acceptF

B$_6$. nonexec !accept ∧ !reject

B$_7$. caused *create*(C(B, S, goods, pay)) if !accept

In two-party protocols, all protocol rules are relevant to each role. In the case of three or more parties, the above naive approach fails. Ignoring other details, assume that the claim receiving protocol *Rec* (Figure 1(r)) includes a rule requiring that an approval be sent only after authOK:

REC$_1$. nonexec approvedA if approvedF ∨ ¬authOKF

The naive approach yields the following for the reporter:

ERR$_1$. nonexec ?approved if ¬authOKF ∨ approvedF

Observe that the reporter is not involved in, and hence not aware of, the interactions involving the messages authOK and authNOK. Thus, rule **ERR$_1$** incorrectly prohibits the receipt of the approved message. The receiver role (reporter in this case) should not depend on the occurrence of messages in which it is not involved. A simple fix is to erase references to messages not visible to the given role. Thus, the correct specification for the reporter would be

RP$_1$. nonexec ?approved if ¬|approved| ∨ approvedF

Notice that **RP$_1$** does not refer to authOKF and authNOKF. The roles are still correct as the sender would send the message only if it was allowed. However, the reporter cannot locally verify whether the call center had received an authorization from the provider when it sent an approval.

Problems of Enactability

Unfortunately, role extraction does not guarantee correctness in the sense of Definition 2: the roles extracted from a protocol $\mathbb{P}$ need not generate conversations that $\mathbb{P}$ generates. This problem cannot be fixed by improving role extraction because it arises when the given protocol is ill-suited for enactment in a distributed setting. We define two classes of problems: *nonlocal choice* and *blindness*. We show below that a protocol yields correct roles iff it is not subject to nonlocal choice and blindness.

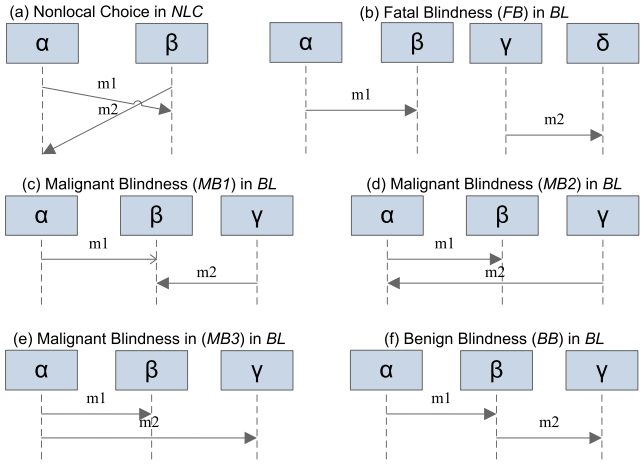

Figure 2: Scenarios: (a) nonlocal choice; (b)–(f) blindness

Nonlocal Choice

Nonlocal choice is common in distributed settings (Ladkin & Leue 1995). To understand the problem of nonlocal choice, consider a protocol *NLC* that specifies that messages $m_1(\alpha, \beta)$ and $m_2(\beta, \alpha)$ cannot happen concurrently.

NLC$_1$. nonexec m_1A ∧ m_2A

Roles (α and β) would be extracted as follows.

NLC-α_1. nonexec $!m_1 \wedge ?m_2$

NLC-β_1. nonexec $?m_1 \wedge !m_2$

However, these roles are incorrect for *NLC*. Because $!m_1$ on α and $!m_2$ on β are both enabled, their mutual exclusivity cannot be guaranteed, thus potentially violating *NLC*, as Figure 2(a) shows.

Blindness

In protocols involving three or more roles, not all roles need necessarily be sufficiently well-informed on the state of the enactment. We call this the problem of *blindness* of roles with respect to certain messages.

To understand blindness, consider a family of protocols *BL* involving the same two messages. *BL* requires that neither m_1 nor m_2 be repeated and that m_2 cannot happen before m_1. Each protocol in the *BL* family pairs the senders and receivers of these messages differently.

BL$_1$. nonexec m_1A if m_1F

BL$_2$. nonexec m_2A if $\neg m_1\text{F} \vee m_2\text{F}$

Benign Blindness Consider protocol *BB* with roles α, β, and γ as in Figure 2(f), which are extracted as:

BB-α_1. nonexec $!m_1$ if m_1F

BB-β_1. nonexec $?m_1$ if $\neg|m_1| \vee m_1\text{F}$

BB-β_2. nonexec $!m_2$ if $\neg m_1\text{F} \vee m_2\text{F}$

BB-γ_1. nonexec $?m_2$ if $\neg|m_2| \vee m_2\text{F}$

Note that γ is blind with respect to m_1 and, as a result, m_1F does not occur in **BB-γ_1**. We can verify by inspection that these roles are correct with respect to *BB*. However, γ has no means to locally verify m_1, and thus to ensure that β sent m_2 only after receiving m_1. For instance, if the agent playing β is faulty, it could send out m_2 without receiving m_1, thus violating *BB*. We call this form of blindness *benign* as the roles are correct but not locally verifiable.

Malignant Blindness Consider protocol *MB1* with roles α, β, and γ as in Figure 2(c), which are extracted as:

MB1-α_1. nonexec $!m_1$ if m_1F

MB1-β_1. nonexec $?m_1$ if $\neg|m_1| \vee m_1\text{F}$

MB1-β_2. nonexec $?m_2$ if $\neg|m_2| \vee \neg m_1\text{F} \vee m_1\text{F}$

MB1-γ_1. nonexec $!m_2$ if m_2F

Here, α and γ are blind with respect to m_2 and m_1, respectively. Thus, there is a possibility that γ sends m_2 before α sends m_1. However, β would not receive m_2 without receiving m_1. The correctness of these roles depends on how a message order requirement in a protocol translates to the requirements in roles. A protocol requirement that m_1 must happen before m_2 can be interpreted in the roles as (SS) *send before send:* $!m_1$ before $!m_2$; (SR) *send before receive:* $!m_1$ before $?m_2$; (RS) *receive before send:* $?m_1$ before $!m_2$; or (RR) *receive before receive:* $?m_1$ before $?m_2$. Although RS is the safest interpretation—roles satisfying it satisfy the other three—each of these interpretations is suitable in different applications. For example, in a purchasing scenario, a shipper must have received a ship order from the merchant before the shipper receives a status request from the customer (RR). *MB1* roles are incorrect if SS or RS is adopted. We call this form of blindness *malignant* as the correctness of the roles depends on the translation of a global requirement to local requirements, which may be contextual. Likewise, *MB2* (Figure 2(d)) is incorrect under SS, RS, or RR. And, *MB3* (Figure 2(e)) is incorrect under RS or RR. This paper treats RS as the standard of correctness. In some applications, the remaining (more flexible) interpretations may be acceptable.

Fatal Blindness Consider protocol *FB* with roles α, β, and γ as in Figure 2(b), which are extracted as:

FB-α_1. nonexec $!m_1$ if m_1F

FB-β_1. nonexec $?m_1$ if $\neg|m_1| \vee m_1\text{F}$

FB-γ_1. nonexec $!m_2$ if m_2F

FB-δ_1. nonexec $?m_2$ if $\neg|m_2| \vee m_2\text{F}$

Here, no interpretation of the message exchange would guarantee the correctness of the roles. This form of blindness is called *fatal* as it clearly represents an error in the modeling of the original protocol.

The possibilities of Figure 2 assume that there are no other rules in the protocol. Adding new rules may correct the roles. For example, in *FB*, a new message $m_3(\beta, \gamma)$ and additional constraints such that m_1 precedes m_3 and m_3 precedes m_2 would preclude fatal and malignant blindness by effectively enforcing an order between m_1 and m_2. Our method accounts for such additional rules.

Proposition 1 *Figures 2(b)–(e) are the only problematic protocols in the family BL.*

Proof Sketch. *BL specifies that message m_1 precedes m_2. There can only be five combinations of three distinct roles being the sender and receiver of two messages m_1 and m_2 where m_1 precedes m_2. We can verify by inspection that Figures 2(b)–(f) show all of these. We showed above that Figure 2(f) is not problematic and Figures 2(b)–(e) are problematic.* ∎

Enactable Protocols

Given the above problematic possibilities in protocols, it is essential to define a set of desirable properties that guarantee enactability. Below $\mathbb{P}$ is a protocol.

Definition 3 $\mathbb{P}$ is *enactable* if (1) the roles extracted from it are guaranteed to be correct, (2) the role KBs are guaranteed to agree on their commitments, and (3) there is at least one conversation of $\mathbb{P}$ in which all commitments created within the protocol are discharged.

Definition 4 *Subscription.* A role α is *subscribed* to

- A commitment operation *op* iff (1) if *op* is *create*, *cancel*, or *release*, then α is the debtor or the creditor of the commitment being manipulated, (2) if *op* is *delegate*, then α is the delegator or the delegatee, and (3) if *op* is *assign*, α is the assigner or the assignee.
- A commitment condition (consequent or antecedent) *cond* of a commitment of which α is a debtor or a creditor.

Definition 5 A message m_j *proves* an occurrence of a message m_i if and only if $\mathbb{P}$ includes a rule of the form ⌜nonexec m_jA if $\neg m_k\text{F} \vee \psi$⌝, and either (1) m_i=m_k or (2) m_k proves an occurrence of m_i.

Definition 6 *Commitment awareness.* A role α is *aware* of

- A commitment operation *op* iff $\mathbb{P}$ includes a rule of the form ⌜caused *op* if ψ⌝ such that α is aware of ψ.
- A commitment condition *cond* iff $\mathbb{P}$ includes a rule of the form ⌜caused *cond* if ψ⌝ such that ρ_i is aware of ψ.

Definition 7 *Message awareness.* A role α is

- *Directly aware* of a message action m_iA and fluent m_iF iff α is the sender or the receiver of m_i.
- *Indirectly aware* of a message m_i via a message m_j iff α is not *directly aware* of m_i but is *directly aware* of m_j and m_j *proves* m_i.
- *Aware* of a message m_i iff α is either *directly aware* of m_i or *indirectly aware* of m_i via m_j.

Definition 8 *Logical awareness.* A role α is *aware* of $\top$ and $\bot$. In the syntax trees generated from the productions of ψ and ψ_f, α is *aware* of a node if α is aware of all its children.

Proposition 2 *α is aware of ψ iff α is aware of $\neg\psi$.* ∎

Based on the above concepts, we state the following properties, which a protocol must satisfy in order to be enactable.
P1. [Localization of choice] If $\mathbb{P}$ includes a rule of the form $\ulcorner$nonexec $m_1A(\alpha,\beta) \wedge m_2A(\gamma,\delta)\urcorner$, and m_1 does not prove m_2, and m_2 does not prove m_1, then $\alpha = \gamma$. That is, any mutually exclusive messages that are not ordered by $\mathbb{P}$ have the same role as their sender (thus the choice between the messages is local). For example, if in addition to the above rule, $\mathbb{P}$ specifies $\ulcorner$nonexec m_1A if $\neg m_2$F$\urcorner$, then m_1 proves m_2. Thus there is no nonlocal choice and the senders of m_1 and m_2 can be different.
P2. [Message verifiability] If $\mathbb{P}$ includes a rule of the form $\ulcorner$nonexec $m_1A(\alpha,\beta)$ if $\psi_f\urcorner$, then the sender α of m_1 is aware of ψ_f. This precludes malignant and fatal blindness. As the receiver β is not subjected to this restriction, benign blindness is allowed.
P3. [Commitment operation verifiability] If a role α is subscribed to a commitment operation *op*, then α is aware of *op*. This precludes protocols that operate on a commitment of α via a message exchange of which α is not aware.
P4. [Commitment condition verifiability] If a role α is subscribed to a condition *cond*, then α is aware of *cond*. This precludes protocols that cause consequents or antecedents of commitments of α via message exchanges of which α is not aware.
P5. [Commitment discharge] For every commitment consequent *cons* in $\mathbb{P}$, the transition system that is the model for $\mathbb{P}$ includes a transition T that occurs on some path such that $T \models cons$. This ensures that all the commitments created in $\mathbb{P}$ can potentially be discharged—in terms of the protocol we cannot assert if an agent playing a role will in fact discharge its commitments, but the protocol allows that. Commitment discharge is necessary to ensure the conclusion of a business transaction, as pending commitments imply unfinished business. For example, *Ord* does not satisfy this property as the commitments created by quote and accept cannot be discharged within the protocol.

Proposition 3 *A protocol $\mathbb{P}$ is enactable iff it satisfies properties P1–P5.*
Proof Sketch. *For the "only if" direction, assume that $\mathbb{P}$ is enactable. By definition, the roles extracted from $\mathbb{P}$ are guaranteed to be correct. Thus, the roles do not exhibit nonlocal choice, malignant, or fatal blindness. If nonlocal choice does not occur, then (1) there is no rule $\ulcorner$nonexec $m_1A(\alpha,\beta) \wedge m_2A(\gamma,\delta)\urcorner$, or (2) α=γ, or (3) concurrent transmission of m_1 and m_2 is prevented due to other rules. Hence, $\mathbb{P}$ satisfies P1. If fatal and malignant blindness do not occur, then for each rule constraining message orders, the sender of the message is either involved in the messages in the premises of the rule, or it can prove the messages in the premises of the rule. Hence, $\mathbb{P}$ satisfies P2. Also by definition, the agents playing roles agree on their commitments. Thus, all agents playing roles are aware of the commitments to which they are subscribed. Hence, $\mathbb{P}$ satisfies P3 and P4. Similarly, for discharge of all the commitments created, $\mathbb{P}$ must satisfy P5. Hence, $\mathbb{P}$ satisfies properties P1–P5.*

For the "if" direction, assume that $\mathbb{P}$ satisfies the properties P1–P5. Due to P1, if concurrent occurrence of messages m_1 and m_2 is prohibited by $\mathbb{P}$, then either the same role is the sender of both, or m_1 and m_2 are strictly ordered due to other rules of $\mathbb{P}$. Thus, $!m_1$ and $!m_2$ cannot happen concurrently. Thus, the problem of nonlocal choice does not occur. Similarly, due to P2, the senders of all messages are aware of the premises of the rules constraining the messages. Thus, the problems of fatal and malignant blindness do not occur. Due to Proposition 1, there are no other problems that can occur. Hence, the roles are guaranteed to be correct. Also, due to P5, in at least one model of $\mathbb{P}$, all commitments created within $\mathbb{P}$ are discharged. Similarly, due to P3 and P4, all debtors and creditors of commitments are aware of the premises of the rules causing antecedents, consequents, or operations of the commitments. Thus, the debtors and creditors are aware of all commitment updates. Hence, $\mathbb{P}$ meets all three requirements of enactability. ∎

The above definitions consider two messages at a time for nonlocal choice and blindness. Since blindness relates to the flow of information about messages, the above definitions apply to blindness with respect to multiple messages, which we can consider separately. However, for nonlocal choice, situations with three or more messages cannot be mapped to choice with two messages at a time. Extending the approach to such situations would be conceptually straightforward but would require considering sets of messages. In general, protocols that impose such constraints involve excessive synchronization: conventionally realized via approaches such as two-phase commit and not practical in loosely coupled distributed settings (Singh and Huhns 2005).

Role Extraction Algorithm

Determining the enactability of a protocol is straightforward. Each of P1–P5 can be checked according to its definition. For brevity, we do not present a separate algorithm. Algorithm 1 extracts roles from enactable protocols.

First, all the rules constraining occurrences of messages in which α is involved are gathered. Next, for all rules in which α is the receiver, premises of which α is not aware are removed (replaced with true in conjunctions and with false in disjunctions). If α is the sender and it is indirectly aware of a premise, it is replaced with the message fluent via which α is indirectly aware of the premise. The resulting set of rules is the desired role.

Algorithm 1: deriveRole($\mathbb{P}$, α): Extracts α from $\mathbb{P}$

```
1  α.R ← getRelevantRules(ℙ, α);
2  foreach r∈α.R ∧ r=⌜nonexec m_i A(β, α) if ψ⌝ do
3      foreach atom ∈ψ do
4          if α is not aware of atom then
5              Remove atom from ψ;
6  foreach r∈α.R ∧ r=⌜nonexec m_i A(α, β) if ψ⌝ do
7      foreach atom ∈ψ do
8          if α is indirectly aware of atom via m_k then
9              Replace atom by m_k;
10 return α.R;
11 Procedure getRelevantRules(ℙ, α): Get rules for α
12 R ← { }; flag=false;
13 foreach r∈ℙ.R do
14     if r=⌜nonexec m_i A(α, β) if ψ⌝ then
15         Replace m_i A with !m_i; flag=true;
16     if r=⌜nonexec m_i A(β, α) if ψ⌝ then
17         Replace ψ with ψ ∧ ¬|m_i|;
18         Replace m_i A with ?m_i; flag=true;
19     if r=⌜op if ψ⌝ ∧ α is subscribed to op then
20         flag=true;
21     if r=⌜cond if ψ⌝ ∧ α is subscribed to cond then
22         flag=true;
23     if flag=true then
24         Replace m_j A with !m_j or ?m_j in ψ if α is directly
           aware of m_j;
25         R ← R ∪ {r};
26 return R;
```

Discussion

Endriss *et al.* (2003) and Baldoni *et al.* (2005) check if a conversation generated by a set of roles does not get "stuck" and conforms to a protocol. Similarly, Chopra & Singh (2008) study a business-level interoperability of a set of roles. These approaches do not address the challenge of deriving conformant roles from a protocol.

Bultan *et al.* (2006) model roles as finite state automata and consider the synchronizability and realizability of their asynchronous conversations. They formally verify whether a particular conversation is realizable via the interactions specified in the automata. They assume that a global conversation specifies orderings between the send events of the messages and not the messages themselves. However, practical applications may impose any of the four possible orderings of sends and receives as discussed above.

Desai & Singh (2004) and Khalaf (2007) extract roles from protocols but consider only two-party protocols. They neither identify nor address nonlocal choice and blindness. Yang *et al.* (2006) seek to formalize WS-CDL, but ignore the above challenges and make strong assumptions such as that all roles share a global state. Busi *et al.* (2006) define a notion of conformance of local orchestrations with a global choreography. However, their notion of conformance is quite strong due to its basis on bisimulation. Enactability is a more flexible and suitable requirement due to its basis in commitments and a flexible communication model.

Zaha *et al.* (2006) identify and target a quite similar problem. Their notion of "local enforceability" of choreographies is similar to the notion of enactability of protocols. However, they address only one of the six forms of blindness (fatal) we identified in this paper. They allow nonlocal choice which they call "two-way inhibits" if the actors of both messages are the same. Figure 2(a) shows how this may result in incorrect roles.

The main contributions of this paper are the necessary and sufficient conditions for enactability of generic, declaratively specified, protocols and a comprehensive treatment of problematic possibilities including multiparty protocols.

This paper fits into our ongoing program of research into formal, semantic bases for interaction. This work will be incorporated into tools for protocol design.

Thanks to Amit Chopra and the reviewers for helpful comments and suggestions.

References

Baldoni, M., Baroglio, C., Martelli, A., and Patti, V. 2005. Verification of protocol conformance and agent interoperability. *CLIMA VI*, 265–283.

Bultan, T., Su, J., and Fu, X. 2006. Analyzing conversations of web services. *Internet Computing* 10(1):18–25.

Busi, N., Gorrieri, R., Guidi, C., Lucchi, R., and Zavattaro, G. 2006. Choreography and orchestration conformance for system design. *COORDINATION*, 63–81.

Chopra, A. K. and Singh, M. P. 2008. Constitutive interoperability. In *AAMAS*. To appear.

Desai, N. and Singh, M. P. 2004. Protocol-based business process modeling and enactment. *ICWS*, 35–42.

Desai, N., Chopra, A. K., Arrott, M., Specht, B., and Singh, M. P. 2007. Engineering foreign exchange processes via commitment protocols. *SCC*, 514–521.

Endriss, U., Maudet, N., Sadri, F., and Toni, F. 2003. Protocol conformance for logic-based agents. *IJCAI*, 679–684.

Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence* 153(1-2):49–104.

Khalaf, R. 2007. From RosettaNet PIPs to BPEL processes. *Data and Knowledge Engineering* 61(1):23–38.

Ladkin, P. B. and Leue, S. 1995. Interpreting message flow graphs. *Formal Aspects of Computing* 7(5):473–509.

Malone, T. W., Crowston, K., and Herman, G. A., eds. 2003. *Organizing Business Knowledge*. MIT Press.

Singh, M. P. and Huhns, M. N. 2005. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley.

Singh, M. P. 1999. An ontology for commitments in multiagent systems. *Artificial Intelligence and Law* 7:97–113.

Yang, H., Zhao, X., Qiu, Z., Pu, G., and Wang, S. 2006. A formal model for web service choreography description language (WS-CDL). *ICWS*, 893–894.

Zaha, J. M., Dumas, M., Hofstede, A. T., Barros, A., and Decker, G. 2006. Service interaction modeling. *EDOC*, 45–55.