ABSTRACT

CHOPRA, AMIT KHUSHWANT. Commitment Alignment: Semantics, Patterns, and Decision Procedures for Distributed Computing. (Under the direction of Professor Munindar P. Singh).

Current service-oriented architectures lack business-level software abstractions. As a result, existing implementations are unnecessarily rigid. This dissertation takes as its point of departure the idea that agents who offer and consumer business services enter into commitments with one another. These commitments give meaning to the interaction among the agents. Commitments support a semantic notion of compliance and enable flexible enactment of business processes.

Interoperability refers to the ability of agents to engage in interaction with one another. In open systems, where agents are autonomous and heterogeneous, ensuring interoperability is critical. Traditionally, interoperability has been formulated in low-level terms. This dissertation presents *commitment alignment* as a key form of business-level interoperability. Agents are aligned if whenever the creditor of a commitment infers the commitment, the debtor infers it too. A misalignment precludes any possibility of successful engagement among agents—their interaction would break down.

This dissertation formally characterizes alignment in multiagent settings. It presents the causes of misalignment, namely, *autonomy*, *distribution*, and *heterogeneity*. Autonomy means that agents communicate asynchronously; distribution refers to the fact that in distributed systems, some agents may have more information than others; and heterogeneity refers to the fact that agents may have incompatible interfaces.

To address autonomy and distribution, we propose a formalization of commitments that consists of three elements: a semantics of the commitment operations; messaging patterns that implement the commitment operations; and weak constraints on agents' behaviors to ensure the propagation of vital information. The constraints result in messages that are critical to alignment. We prove that under our formalization, no misalignment occurs because of autonomy or distribution. To address heterogeneity, we propose a language for agent interfaces, formulate a decision procedure that checks for interface compatibility, and prove its correctness. By addressing all three causes, we guarantee that no misalignment occurs.

Commitment Alignment: Semantics, Patterns, and Decision Procedures for Distributed Computing

by
Amit Khushwant Chopra

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina

2008

APPROVED BY:

Dr. Rada Y. Chirkova	Dr. S. Purushothaman Iyer
Dr. Mladen A. Vouk	Dr. Munindar P. Singh Chair of Advisory Committee

DEDICATION

To my mother

BIOGRAPHY

Amit was born in Mumbai to Asha and Khushwant Chopra. His family moved to Pune sometime in the late eighties in pursuit of a better life.

Amit obtained his B.E. in Computer Engineering from the Pune Institute of Computer Technology in 1999. Until July 2001, he worked at Persistent Systems (Pune) as a software engineer. Amit then moved to the United States to pursue graduate studies in Computer Science at North Carolina State University. In 2006, he took a break from graduate studies and poverty to work at IBM's WebSphere Technology Institute. In January 2009, Amit will join the University of Trento (Italy) as a postdoctoral researcher.

Amit is interested in distributed computing, agent-oriented software engineering, and service-oriented architectures. He likes reading novels and short stories, watching films (*strictly* in the comfort of his home), swimming, running, and playing cricket, table tennis, and soccer.

ACKNOWLEDGMENTS

I am deeply indebted to my advisor Munindar Singh. Munindar read countless drafts, corrected numerous mistakes, and provided astute guidance. As I move on to newer pastures, I hope I have imbibed some of Munindar's enthusiasm, work ethic, vision, and generosity.

I am grateful to Professors Chirkova, Iyer, and Vouk for the time and effort they expended as my committee members. Over the years, I have benefitted amply from their questions and suggestions.

I have benefitted greatly from interactions with former and current colleagues at NC State, especially Nirmit Desai (IBM Research) and Ashok Mallya (eBay). Scott Gerard and Pankaj Telang have happily served as sounding boards for my ideas. Chris Hazard provided useful comments on various drafts and presentations.

I have had several interesting discussions on agent interoperability with Matteo Baldoni, Cristina Baroglio, and Viviana Patti (all at the University of Torino). Michael Winikoff (University of Otago) and Gal Kaminka (Bar Ilan University) gave me useful advice, technical and professional. I also had fruitful discussions about my work with Annie Antón, James Lester, Laurie Williams, and Tao Xie (all at NC State).

A special thanks goes out to the staffs of the Department of Computer Science and the Office of International Services at NC State. They made all administrative procedures relatively painless.

Ashok Mallya, Mrudula Neginhal, Vishwas Puttasubbappa, Hemant Ramnani, and Amit Sharma gave me the gift of unwavering friendship. For that, I'm deeply grateful.

I have been away from my parents for a long time now. My mother, especially, has felt the pangs of my absence. I owe her much.

Soon I will be leaving the shores of the United States, and I do not know if I will ever have the opportunity to come live here again. I feel a sense of gratitude towards the people of the United States—living here has been a singularly liberating experience.

This research was partially supported by the National Science Foundation under grant IIS-0139037, by DARPA under contract F30603-00-C-0178, by an IBM Faculty Award, and by a gift from Intel. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction: Commitments

Today, the world is witnessing an explosive growth in the number and richness of online services. We use such services daily—when we are pay our utility bills, make airline and hotel reservations, buy and sell things on eBay, manage our bank accounts ...the list is endless. The growth is not limited to consumer facing services; more and more organizations are conducting their procurement and sales processes electronically. The proliferation of such services belies the complexity of their construction. Most service enactments involve multiple autonomous and heterogeneous organizations. For example, booking an airline ticket involves not just the user, the travel agency, and the airline, but also various credit card companies and banks, each with their own independently designed information systems. This raises the tremendous challenge of interoperation: how can we ensure interoperation between such diverse business organizations?

The question of interoperability squarely puts the focus on the modeling of interactions among organizations. In open settings, such as the Web, accommodating flexible interaction is as much a concern as interoperability. From a business point of view, it makes sense that an organization offering services on the Web should be able to interact and conduct business with the *widest* possible set of organizations, and be able to take advantage of opportunities and handle exceptions. However, there is a certain tension between interoperability and flexibility. Flexibility means that more interactions are supported, which makes determining interoperability tougher.

This chapter shows that current approaches focus solely on interoperation and fare poorly when it comes to accommodating flexibility. We then motivate an approach that

supports flexible interactions. Our approach recognizes the lack of business-level abstractions in current approaches for modeling interactions, and fills that gap. This chapter lays the groundwork for the results in this dissertation.

1.1 Trends in Business Processes

Traditionally business processes were used to automate the internal operations of an organization. The operations were modeled as workflows. Workflow management systems [Georgakopoulos et al., 1995] such as MQSeries, Lotus Notes, FlowMark, and Staffware were used to enact and manage the flows. This was before the e-business era, and the typical applications were form processing, office automation, computer-supported cooperative work, and the paperless office.

As the potential of e-business became apparent, organizations sought not only to automate their internal operations, but also to automate interactions with their business partners. Interoperation posed the biggest challenge: how do heterogeneous software components representing different organizations interact meaningfully? That gave rise to approaches such as EDI (Electronic Data Interchange). The EDI standard describes data formats for documents such as a purchase order. The business organizations participating in an EDI process agree upon the documents to be transmitted and how they should be used. Automated supply chains emerged in this stage. The problem with EDI, however, was that it resulted in preconfigured, highly customized, and inflexible processes. Instead of achieving interoperation among the participants, EDI resulted in integration of the participants' information systems—thoroughly less desirable.

Web services emerged to address the problem of heterogeneity in a more elegant way. A Web service is an application with a published interface that clients can use to interact with it. Interfaces are the basis of interoperation; a client has simply to act according to the interface to avail of the Web service. Two principal approaches for weaving Web services into business processes emerged: *orchestration* and *choreography*.

1.1.1 Orchestration

Orchestration refers to the coordinated invocation of services from a workflow. The constructs used for coordinating the services are primary those from structured programming—

loops, conditional branching, fork, join, and so on. BPEL [2003] is a widely used and industry-supported language for writing such workflows. OWL-S [Martin et al., 2007] is another language designed for orchestrating services, with a special emphasis on Semantic Web. BPMN [2008] is a graphical notation for expressing orchestrations. Orc [Cook and Misra, 2007] is a minimal orchestration language with only three coordination primitives using which complex orchestrations may be built.

Figure 1.1 shows the workflow of a merchant in BPMN notation. The merchant receives an order from a customer, following which the merchant invokes the Process Order service. After the order is processes, the merchant invokes the Check Inventory and Process Payment services in parallel. If the item ordered is not in the inventory, the merchant checks again. Otherwise, the item is shipped to the customer. If the payment is processed as well, the merchant sends a confirmation to the customer and exits the workflow.

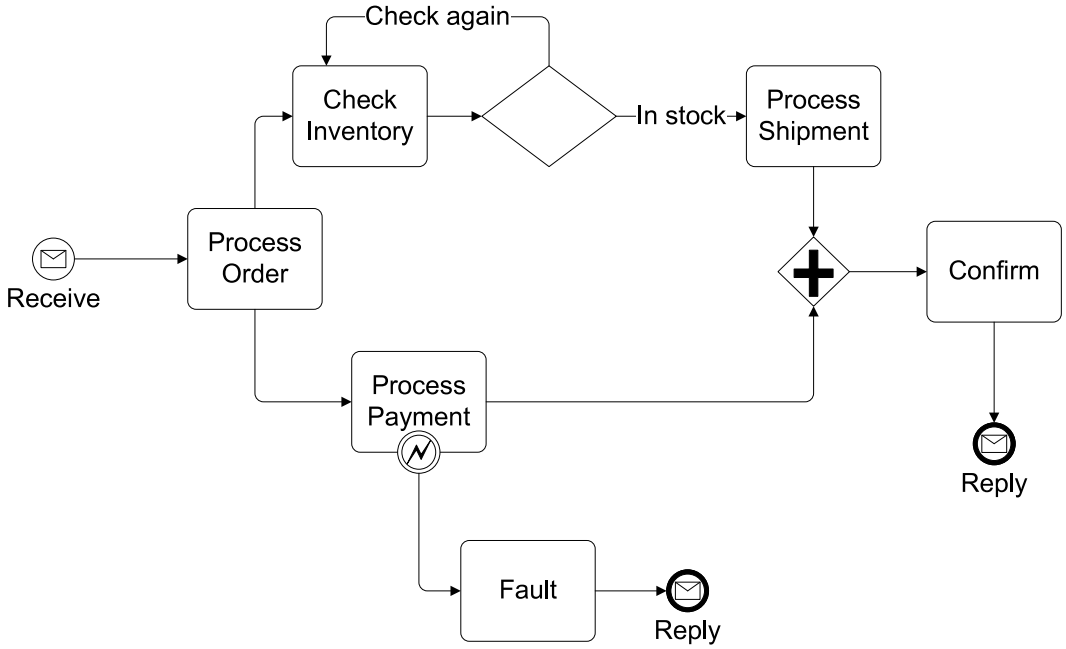

Figure 1.1: A merchant's business process, modeled in BPMN

We make the following observations about orchestration-based approaches.

- An orchestration reflects only one participant's view of the overall business process. Figure 1.1, for example, reflects only the merchant's point of view—*how* the merchant

processes an order. An orchestration is, in fact, the *implementation* of a particular participant. Notice how the internal policies of merchant, as indicated by the decision node representing the inventory check, is mixed with service invocations. Whether to invoke services concurrently or serially is also an internal policy matter for the merchant.

- Orchestration is an invocation-based approach. Services are treated on par with computational objects. In Figure 1.1, the merchant invokes the Process Order, Check Inventory, Process Payment, and Process Shipment services to fulfill an order. Interactions with other participants in the overall business process find no place in an orchestration. For example, in Figure 1.1, participants that one might reasonably expect to be involved in an order fulfillment process, such as a bank and a shipper, find no place. Not just that, even the customer is not explicitly represented. True, such participants may be behind the services that are invoked. However, a service is not the same as a participant. Participants have autonomy and can be engaged, whereas services are merely invoked.

1.1.2 Choreography

A choreography *prescribes* how the participants in a business process interact by specifying the flow of messages between them. This is achieved by using control and data flow constructs, not unlike those used for specifying orchestrations.

A choreography may be understood as the interaction protocol among the participants. Typically, a choreography is specified in terms of *roles* rather the participants themselves. Participants *adopt* roles, that is, bind to the roles, in the choreography. This promotes reusability.

WS-CDL [2005] and ebBP [2006] are the leading industry supported choreography standardization efforts. UML [Huget and Odell, 2005] is another leading approach for specifying choreography. A choreography may also be specified formally: as state machines [Yellin and Strom, 1997; Benatallah et al., 2004], Petri Nets [Cost et al., 1999], statecharts [Dunn-Davies et al., 2005], processes in the π-calculus [Canal et al., 2003], or via more declarative approaches [Singh, 2003; van der Aalst and Pesic, 2006].

Figure 1.2 shows a choreography between two roles, merchant (mer) and customer

(cus) as a state machine. The transitions are labelled with messages; the prefix mer,cus indicates a message from the merchant to the customer, and cus,mer indicates a message from the customer to the merchant. This choreography supports two executions. One execution represents the scenario where the customer rejects the merchant's offer. The other execution represents the scenario where the customer accepts the offer, following which the merchant and the customer exchange the item and the payment for the item.

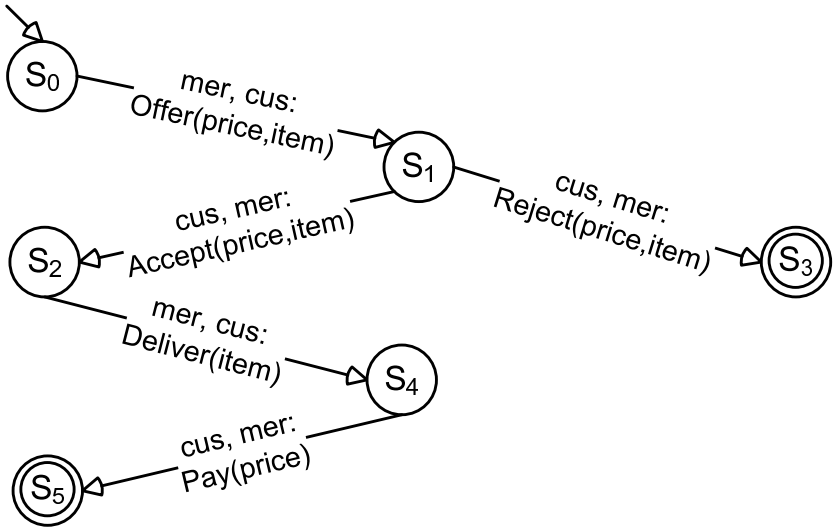

Figure 1.2: A choreography specified as a state machine

We make the following observations about choreographies.

- A choreography specifies only the interactions among participants. In doing so, it abstracts away from some of the implementation details of the participants. For example, Figure 1.2 does not reflect the internal policies based upon which the customer accepts an offer. Further, a choreography specifies the interactions of *all* the participants in a business process.

- Choreography is an interaction-oriented approach. It acknowledges autonomy by giving first-class status to participants and their interactions.

The trend lately is to use choreographies as modeling abstractions for business processes as evidenced by the growing number of such specifications [RosettaNet, 1998;

HL7; TWIST]. We find this trend encouraging; however, this approach also falls short in supporting autonomy.

Consider the choreography in Figure 1.3. The dotted paths indicate two additional executions that are not supported by the choreography in Figure 1.2. The executions depict the scenarios where the customer sends the payment upon receiving an offer and after sending accept, respectively. These additional executions are just as sensible as the original ones. However, in the context of the choreography in Figure 1.2, these executions are trivially *noncompliant*. The reason is that checking compliance with choreographies is purely syntactical—the messages have to flow between the participants exactly as prescribed. Clearly, this curbs the participants' autonomy and flexibility.

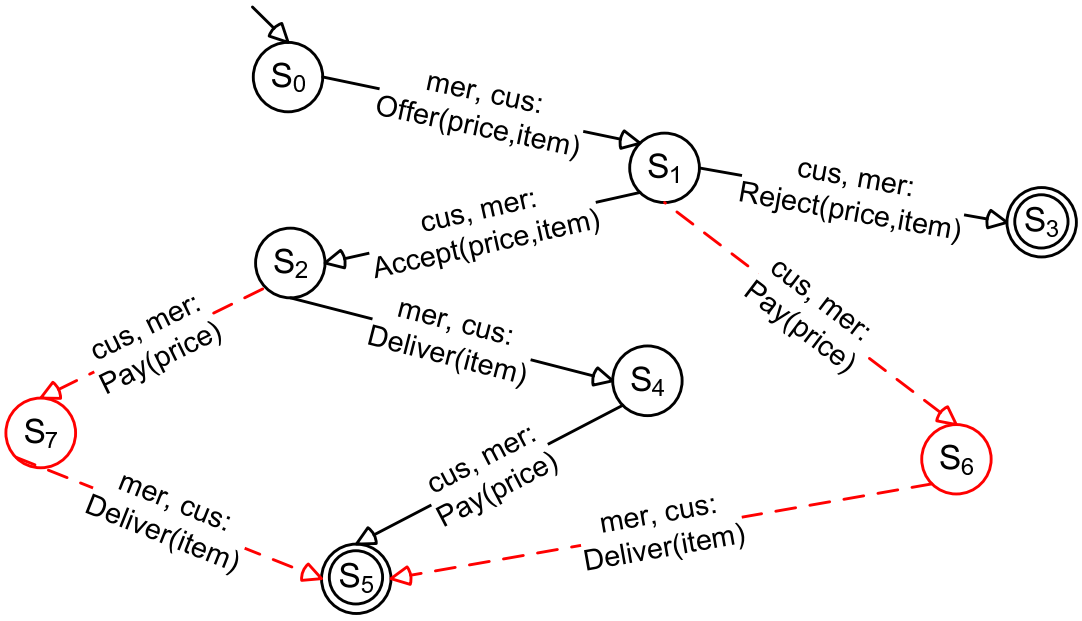

Figure 1.3: An alternative, more flexible choreography

1.2 Commitment Protocols

In contrast with choreography and orchestration-based approaches, commitment protocols gives primacy to the *business meanings* of service engagements, which are captured through the participants' *commitments* to one another [Yolum and Singh, 2002],

[Chopra and Singh, 2004; Singh et al., 2004; Desai et al., 2005; Winikoff et al., 2005], [Desai et al., 2007b]. Computationally, each participant is modeled as an *agent*; interacting agents carry out a service engagement by creating and manipulating commitments to one another.

1.2.1 Commitments

A commitment is of the form $\mathsf{C}(debtor, creditor, antecedent, consequent)$, where *debtor* and *creditor* are agents, and *antecedent* and *consequent* are propositions. A commitment $\mathsf{C}(x, y, r, u)$ means that x is committed to y that if r holds, then it will bring about u. If r holds, then $\mathsf{C}(x, y, r, u)$ is *detached*, and the commitment $\mathsf{C}(x, y, \top, u)$ holds ($\top$ being the constant for truth). If u holds, then the commitment is *discharged* and doesn't hold any longer. All commitments are *conditional*; an unconditional commitment is merely a special case where the antecedent equals $\top$. Examples 1–3 illustrate these concepts. In the examples, Bookie is a bookseller, and Alice is a customer.

Example 1 (Commitment) $\mathsf{C}(Bookie, Alice, \$12, BeatingtheOdds)$ means that Bookie commits to Alice that if she pays \$12, then Bookie will sends her the book *Beating the Odds*. ▮

Example 2 (Detach) If Alice makes the payment, that is, if \$12 holds, then $\mathsf{C}(Bookie, Alice, \$12, BeatingtheOdds)$ is detached. In other words, $\mathsf{C}(Bookie, Alice, \$12, BeatingtheOdds) \wedge \$12 \Rightarrow \mathsf{C}(Bookie, Alice, \top, BeatingtheOdds)$. ▮

Example 3 (Discharge) If Bookie sends the book, that is, if *BeatingtheOdds* holds, then both $\mathsf{C}(Bookie, Alice, \$12, BeatingtheOdds)$ and $\mathsf{C}(Bookie, Alice, \top, BeatingtheOdds)$ are discharged. In other words, $BeatingtheOdds \Rightarrow \neg\mathsf{C}(Bookie, Alice, \$12, BeatingtheOdds) \wedge \neg\mathsf{C}(Bookie, Alice, \top, BeatingtheOdds)$. ▮

Importantly, commitments can be manipulated, which supports flexibility. The commitment operations are reproduced below (from [Singh, 1999]). CREATE, CANCEL, and RELEASE are two-party operations, whereas DELEGATE and ASSIGN are three-party operations.

- CREATE(x, y, r, u) is performed by x, and it causes $\mathsf{C}(x, y, r, u)$ to hold.

- CANCEL(x, y, r, u) is performed by x, and it causes $\mathsf{C}(x, y, r, u)$ to not hold.
- RELEASE(x, y, r, u) is performed by y, and it causes $\mathsf{C}(x, y, r, u)$ to not hold.
- DELEGATE(x, y, z, r, u) is performed by x, and it causes $\mathsf{C}(z, y, r, u)$ to hold.
- ASSIGN(x, y, z, r, u) is performed by y, and it causes $\mathsf{C}(x, z, r, u)$ to hold.

We introduce INFORM(x, y, r) as an operation performed by x to inform y that the r holds. It is not a commitment operation, but may indirectly affect commitments by causing detaches and discharges. In relation to Example 2, when Alice informs Bookie of the payment by performing INFORM$(\mathit{Alice}, \mathit{Bookie}, \$12)$, then the proposition \$12 holds, and causes a detach of $\mathsf{C}(\mathit{Bookie}, \mathit{Alice}, \$12, \mathit{BeatingtheOdds})$. A complete, formal treatment of commitments is presented in Chapter 2.

A commitment arises in a social or legal context. The context defines the rules of encounter among the interacting parties, and often serves as an arbiter in disputes and imposes penalties on parties that violate their commitments. For example, eBay is the context of all auctions that take place through their service; if a bidder does not honor a payment obligation for an auction that it has won, eBay may suspend the bidder's account.

1.2.2 Protocol Specification

Table 1.1 shows the specification of a commitment protocol between a merchant and a customer (omitting sort and variable declarations). It simply states what the meanings of the messages are in terms of commitments between the merchant and customer. For instance, the message $\mathit{Offer}(\mathit{mer}, \mathit{cus}, \mathit{price}, \mathit{item})$ means the creation of the commitment $\mathsf{C}(\mathit{mer}, \mathit{cus}, \mathit{price}, \mathit{item})$, meaning the merchant commits to delivering the item if the customer pays the price; $\mathit{Reject}(\mathit{cus}, \mathit{mer}, \mathit{price}, \mathit{item})$ means a release of the commitment; $\mathit{Deliver}(\mathit{mer}, \mathit{cus}, \mathit{item})$ means that the proposition item holds.

Figure 1.4(A) shows an execution of the protocol and Figure 1.4(B) its meaning in terms of commitments. (The figures depicting executions use a notation similar to UML interaction diagrams. The vertical lines are agent lifelines; time flows downward along the lifelines; the arrows depict messages between the agents; and any point where an agent sends or receives a message are annotated with the commitments that hold at that point. In the Figures, instead of writing CREATE, we write Create. We say that

Table 1.1: A commitment protocol

$Offer(mer, cus, price, item)$ means CREATE$(mer, cus, price, item)$
$Accept(cus, mer, price, item)$ means CREATE$(cus, mer, item, price)$
$Reject(cus, mer, price, item)$ means RELEASE$(mer, cus, price, item)$
$Deliver(mer, cus, item)$ means INFORM$(mer, cus, item)$
$Pay(cus, mer, price)$ means INFORM$(cus, mer, price)$

the *Create* message realizes the CREATE operation. Likewise, for other operations and INFORM.) In the figure, the merchant and customer are played by Bookie and Alice, respectively; c_B and c_{UB} are the commitments $\mathsf{C}(Bookie, Alice, \$12, BeatingtheOdds)$ and $\mathsf{C}(Bookie, Alice, \top, BeatingtheOdds)$ respectively; *BO* is abbreviation for *Beating the Odds*.

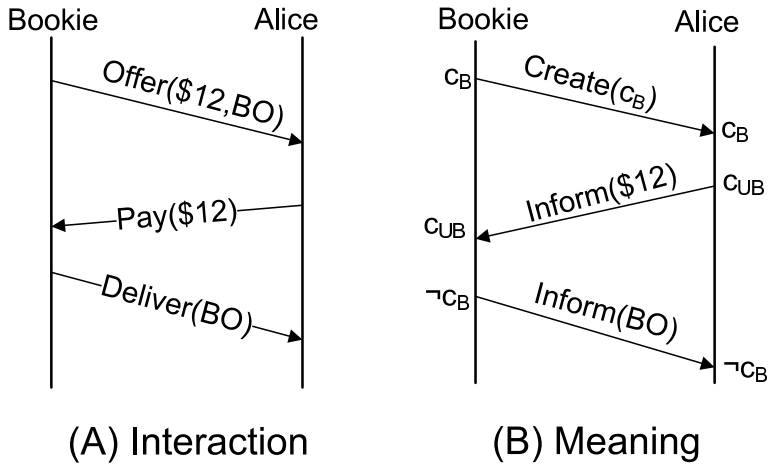

Figure 1.4: Interaction and meaning

1.2.3 Compliance and Flexibility

Service enactments can be judged correct as long as the parties don't violate their commitments. This enhances flexibility over traditional approaches by expanding the operational choices for each party [Chopra and Singh, 2006a]. For example, if the customer substitutes a new way to make a payment or elects to pay first, no harm is done, because the behavior is correct at the business level. And, the merchant may employ a new shipper; the

Table 1.2: Comparison of approaches

	Orchestration	**Choreography**	**Commitment Protocols**
Abstraction	control & data	control & data	business
Compliance	lexical	syntactic	semantic
Flexibility	low	low	high

customer may return damaged goods for credit; and so on. Conversely, a customer would be in violation if he keeps the goods but fails to pay. In this manner, commitments support business-level compliance and don't dictate specific operationalizations [Desai et al., 2005]. By contrast, without business meaning, exercising any such flexibility would result in noncompliant executions.

Figure 1.5 shows some of the possible enactments based on the protocol in Table 1.1. Figure 1.5(B) reflects the execution where the book and payment are exchanged in Figure 1.2; c_A and c_{UA} are $\mathsf{C}(Alice, Bookie, BeatingtheOdds, \$12)$ and $\mathsf{C}(Alice, Bookie, \top, \$12)$, respectively. Figures 1.5(A) and (C) reflect the additional executions supported in Figure 1.3; Figure 1.5(D) reflects a new execution that we hadn't considered before, one where Alice sends an *Accept* even before receiving an offer. All these executions are compliant executions in terms of commitments, and thus supported by the protocol in Table 1.1.

Table 1.2 summarizes the three approaches.

1.3 Dissertation Topic: Commitment Alignment

A challenge that arises in distributed systems is that of state alignment. Alignment is closely related to the problem of consistency in distributed systems. Consistency in a distributed system is typically achieved by synchronizing the nodes in the system. For example, as is common in business process implementations, a two-phase commit protocol is used to synchronize the parties. By alignment, we mean that the agents involved in an interaction should have a consistent view of their commitments to each other. If agents cannot even agree about their commitments to each other, one can hardly expect that they'd be able to engage in business, that is, interoperate with each other.

Informally, we say that agents are aligned, if whenever an agent infers a commit-

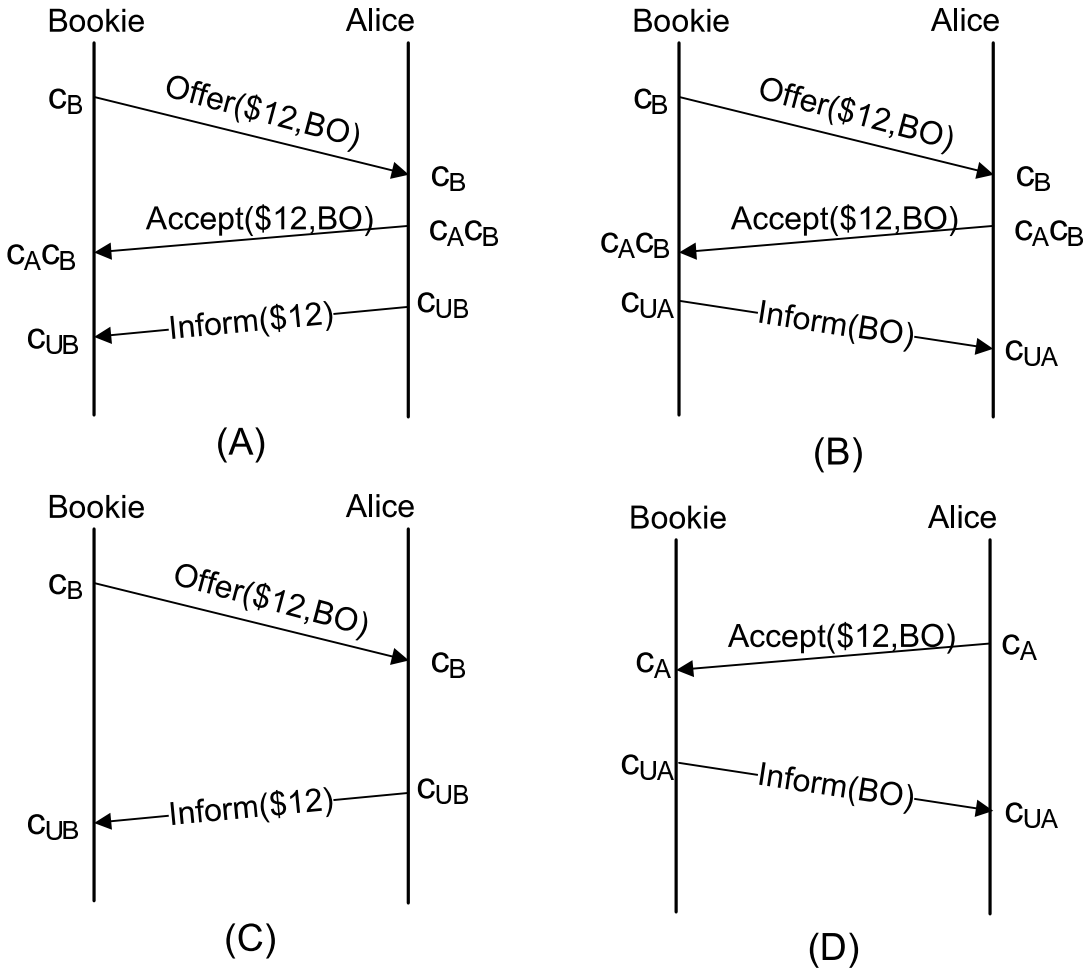

Figure 1.5: Flexible enactment

ment in which it is the creditor, the debtor of the commitment also infers that commitment. Let's consider some examples of misalignment.

Example 4 Bookie sends an *Offer* to Alice, which means a commitment that if Alice pays, then Bookie will send the book. Alice sends the payment (message) for the book. Concurrently, Bookie cancels the offer by sending *CancelOffer*. Alice observes Bookie's cancellation after sending the payment; so she regards it as spurious. Bookie observes Alice's payment after sending cancellation, so Bookie considers the payment late. As a result Alice infers that Bookie is committed to sending her the book, but Bookie does not infer that. Thus, Bookie and Alice are misalignment. ▌

Example 5 Alice commits to Bob that if the sky is clear at 5PM, then she will meet him at the lake. At 5PM, Bob observes (a message from the environment) that the sky is clear, and therefore infers that Alice is unconditionally committed to meeting him at the lake. However, Alice does not know that the sky is clear, and therefore does not infer the unconditional commitment. Bob and Alice are thus misaligned.∎

Example 6 For Alice, an *Offer* message from Bookie counts as a commitment from Bookie to ship a book in return for payment. Whereas for Bookie, *Offer* does not count as any such commitment; but an explicit *Accept* from Alice does. Thus, when Bookie sends Alice an *Offer* message, Alice infers the commitment, but Bookie does not—a misalignment.∎

1.3.1 Causes of Misalignment

There are three possible causes of misalignment.

Autonomy Agents are autonomous. This means that they are free to send messages. In turn, this means that communication between them is asynchronous. Thus, in general, agents observe messages in different orders. If their observations are incompatible, it may cause a misalignment. This is the cause of misalignment in Example 4.

Distribution In a distributed system, some agents may have more information than others. This is the cause of misalignment in Example 5: Bob knows that the sky is clear, but Alice does not.

Heterogeneity Agents may assign incompatible meanings to the messages they are exchanging. To be able to successfully interact, the agents must agree on what their communications count as. Heterogeneity is the cause of misalignment in Example 6.

1.3.2 Results

Our contributions are the following.

1. Commitment alignment is a key form of business-level interoperability. Before engaging in business, parties would want to know if their commitments align suitably. We formally characterize commitment alignment in multiagent systems.

2. We address the challenge of autonomy by defining the semantics of commitment operations. We handle distribution by putting local constraints on the behaviors of agents. Under the semantics and the constraints together, we prove that autonomy and asynchrony cause no misalignments.

3. We delineate two criteria that any approach for alignment must meet: *autonomy compatibility* and *semanticity*. Both autonomy compatibility and semanticity are crucial in arriving at a general solution for alignment. Our formalization meets both these criteria.

4. To handle heterogeneity, we formalize the interfaces of agents in terms of commitments, and present a decision procedure which determines if agents' interfaces are compatible.

If no misalignment happen due to autonomy and distribution, and if the interfaces of agents are compatible (for the interface language we use), then no misalignment occurs.

1.4 Organization

Chapter 2 delineates formally the communication model and agents, and characterizes commitment alignment in terms of this model. Chapter 3 presents a solution to the challenges of autonomy and distribution by formalizing commitment operations and constraints on agent behavior. More importantly, it proves that the solution guarantees that no misalignment occurs due to autonomy or distribution. In Chapter 3, we assume that agents assign identical meanings to messages. In Chapter 4, we drop this assumption, and address heterogeneity. Here, we present a decision procedure to determine interface compatibility. Finally, Chapter 5 places this work in a broader context and outlines future directions.

Chapter 2

Commitment Alignment

We present reasoning postulates for commitments, and a general model of asynchronous communication. Finally, we characterize alignment itself.

2.1 Commitments

Below, x,y, etc are variables over agents; p, q, r, etc. are propositional variables; $\vee$, $\wedge$, $\neg$, $\rightarrow$, $\leftrightarrow$ are the usual propositional connectives; $\top$ and $\bot$ are the constants for truth and falsity, respectively; $\vdash$ is the usual propositional inference symbol. Read $\Rightarrow$ as *implies*.

A commitment is of the form $\mathsf{C}(x, y, r, u)$. If r holds, then $\mathsf{C}(x, y, r, u)$ is *detached*, and the commitment $\mathsf{C}(x, y, \top, u)$ holds. If u holds, then the commitment is *discharged* and doesn't hold any longer. All commitments are *conditional*; an unconditional commitment is merely a special case where the antecedent equals $\top$.

2.1.1 Reasoning Postulates for Commitments

Reasoning postulates for commitments are reproduced below [Singh, 2008]. For brevity, we omit the agents when they can be understood from the context. Further, when the postulates uniformly use the debtor x and creditor y, we write $\mathsf{C}(r, u)$ instead of $\mathsf{C}(x, y, r, u)$.

B1. DISCHARGE. $u \rightarrow \neg\mathsf{C}(r, u)$

B2. DETACH. $\mathsf{C}(r \wedge s, u) \wedge r \rightarrow \mathsf{C}(s, u)$.

B3. AUGMENT. From $\mathsf{C}(r, u) \wedge s \vdash r$, infer $\mathsf{C}(s, u)$

B4. L-DISJOIN. $\mathsf{C}(r, u) \wedge \mathsf{C}(s, u) \rightarrow \mathsf{C}(r \vee s, u)$

B5. R-CONJOIN. $\mathsf{C}(r, u) \wedge \mathsf{C}(r, v) \rightarrow \mathsf{C}(r, u \wedge v)$

B6. CONSISTENCY. $\neg\mathsf{C}(r, \bot)$

B7. NONVACUITY. From $r \vdash u$, infer $\neg\mathsf{C}(r, u)$

B8. WEAKEN. $\mathsf{C}(r, u \wedge v) \wedge \neg u \rightarrow \mathsf{C}(r, u)$

Notice that B1 covers the discharge of commitments. B2 generalizes their detach. *Semanticity* means that alignment must not fail in the face of reasoning postulates B1–B8. That is, we must make sure that the effects of the various messages on commitments are consistent with respect to the above postulates.

2.1.2 Commitment Operations

The commitment operations are reproduced below (from [Singh, 1999]). CREATE, CANCEL, and RELEASE are two-party operations, whereas DELEGATE and ASSIGN are three-party operations.

- CREATE(x, y, r, u) is performed by x, and it causes $\mathsf{C}(x, y, r, u)$ to hold.
- CANCEL(x, y, r, u) is performed by x, and it causes $\mathsf{C}(x, y, r, u)$ to not hold.
- RELEASE(x, y, r, u) is performed by y, and it causes $\mathsf{C}(x, y, r, u)$ to not hold.
- DELEGATE(x, y, z, r, u) is performed by x, and it causes $\mathsf{C}(z, y, r, u)$ to hold.
- ASSIGN(x, y, z, r, u) is performed by y, and it causes $\mathsf{C}(x, z, r, u)$ to hold.

2.1.3 Messages

Let us define the set of messages that agents can exchange. Let Φ be a set of atomic propositions. $\mathit{Inform}(x, y, p)$ is a message from x to y, where p is conjunction over Φ. In the commitment operations, r is a DNF formula over Φ (for example, $(\phi_0 \wedge \phi_1) \vee (\phi_3 \wedge \phi_4)$), and u is a CNF formula over Φ (for example, $(\phi_0 \vee \phi_1) \wedge (\phi_3 \vee \phi_4)$) .

$Create(x, y, r, u)$ and $Cancel(x, y, r, u)$ are messages from x to y; $Release(x, y, r, u)$ from y to x; $Delegate(x, y, z, r, u)$ from x to z; and $Assign(x, y, z, r, u)$ from y to x.

Suppose $c = \mathsf{C}(x, y, r, u)$. Then $Create(c)$ stands for $Create(x, y, r, u)$. We similarly define $Delegate(c, z)$, $Assign(c, z)$, $Release(c, y)$, and $Cancel(c, x)$.

All atomic propositions are stable, that is, if an atomic proposition holds, it holds forever. In English, stability corresponds to the perfective aspect, for example, *book has been delivered*, *payment has been made*, and so on [Singh, 2008]. Propositions with explicit time, such as *the book is delivered by 3PM* are also stable. Thus each atomic proposition corresponds to the occurrence of an *event*: when the proposition holds, the corresponding event is said to have occurred. A commitment, however, is not a stable proposition. A commitment may come to not hold because it was discharged, cancelled, or released, leaving the agents sensitive to race conditions over commitments.

2.1.4 Commitment Strength

We'll use the following commitments frequently.

- $c_B = \mathsf{C}(Bookie, Alice, \$12, BeatingtheOdds)$
- $c_G = \mathsf{C}(Bookie, Alice, \$12, GamblingTips)$
- $c_0 = \mathsf{C}(Bookie, Alice, \$12, BeatingtheOdds \wedge GamblingTips)$
- $c_1 = \mathsf{C}(Bookie, Alice, \$12 \vee coupon, BeatingtheOdds)$
- $c_2 = \mathsf{C}(Bookie, Alice, \$12 \wedge coupon, BeatingtheOdds)$

Intuitively, c_0 is a stronger commitment than c_B (an additional book for the same price); c_1 is stronger than c_B (two ways to obtain a book instead of one); c_B is stronger than c_2 (fewer conditions need to be satisfied to obtain a book). Definition 1 captures this intuition.

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

Thus, for example, $c_0 \succeq c_B$. If $\mathsf{C}(x, y, r, u) \succeq \mathsf{C}(x, y, s, v)$ but $\mathsf{C}(x, y, s, v) \not\succeq \mathsf{C}(x, y, r, u)$, we say $\mathsf{C}(x, y, r, u) \succ \mathsf{C}(x, y, s, v)$. B3 and B8 capture the notion of strength

deductively. For example, if c_1 holds, then by B3, c_B holds as well. Similarly, if c_0 holds, then by B8, c_B holds as well—unless $BeatingtheOdds$ holds already in which case according to B1, c_B cannot hold.

2.2 Agents and Communication

Agents communicate by messaging. Below m, m', m_0, ... are variables over messages. Assumptions A1–A4 model communication.

A1. Communication is *point-to-point*. Below $m(x, y)$ indicates a message m from x to y.

A2. An agent observes all and only those messages that it sends or receives. Observations are ordered *serially*. All observations pertain to messages. Observations of the environment are treated as messages from Env.

A3. Messaging is *reliable*. Messages are neither created nor destroyed by the infrastructure.

A4. Messaging is *ordered*. Any two messages sent by an agent to the same recipient are received in order.

An agent x's *observation sequence* $\langle m_0, \ldots, m_n \rangle_x$ describes the sequence of messages x observes in a particular execution. Let $\mathcal{A}$ be a system of k agents. Then, $O = [O_0, \ldots, O_{k-1}]$ is an *observation vector* over $\mathcal{A}$, where the O_is are the observation sequences, one for each of the k agents. Below, o is a variable over observation vectors; o_x, etc. are variables over a particular agent's observation sequence. A3 and A4 impose validity requirements on vectors.

Definition 2 An observation vector O over $\mathcal{A}$ is *valid* iff $\forall x, y \in \mathcal{A}$: (1) if $m(x, y)$ occurs in O_y, then $m(x, y)$ occurs in O_x; and (2) if $m_1(x, y)$ occurs in O_y, and $m_0(x, y)$ precedes $m_1(x, y)$ in O_x, then $m_0(x, y)$ precedes $m_1(x, y)$ in O_y.

Conditions (1) and (2) in Definition 2 capture A3 and A4, respectively. This paper considers only valid observation vectors.

Think of an agent's observation sequence as representing the agent's state at the granularity of the interaction (i.e., ignoring aspects of the agent's state not reflected in its

observations). Then an observation vector represents the state of the system. $\mathcal{O}_{\mathcal{A}}$, the set of all possible observation vectors for system $\mathcal{A}$, is the set of all possible executions of $\mathcal{A}$.

2.3 Formalizing Alignment

Alignment means that whenever an agent infers a commitment from its observations in which it is the creditor, then the debtor must also infer the commitment from its observations. An execution of a multiagent system is a progression of the system from one (system) state to another. Every time an agent sends or receives a message, the system progresses to a new state. Although we would like to consider all possible executions of the system, we need to ensure that we verify alignment only at well-defined milestones; otherwise, we would falsely claim misalignment. The appropriate milestones are expressed via *quiescence* and *integrity*.

Figure 2.2(B) exemplifies our graphical notation. We represent an execution as a sequence diagram. Each point where a message is sent or received is annotated with the commitments inferred immediately after the observation. If $\mathsf{C}(r, u)$ holds and $\mathsf{C}(r, u) \succeq \mathsf{C}(s, v)$, we only show $\mathsf{C}(r, u)$. Each agent's vertical line may be annotated at the top to indicate initial conditions of the interaction.

2.3.1 Quiescence

A system state is quiescent if no messages are in transit. In considering only quiescent states, we ensure the agents are "synced" up when we verify their alignment. Without quiescence, alignment is generally impossible because some agents may not yet have observed messages destined for them.

Consider Figure 2.1 where Alice has sent $12 to Bookie but Bookie hasn't received the payment. The red line indicates one such point in the interaction. Alice infers that Bookie is now committed to sending her the book, but Bookie has no clue of an incoming payment, and so isn't committed. At quiescence—one such point is indicated by the green line—Bookie would have received the payment. If even at quiescence, Alice and Bookie disagree, we have a problem on our hands. Quiescence may only be temporary, because the agents could be silently computing: it would end when an agent sends a message based on its internal computations.

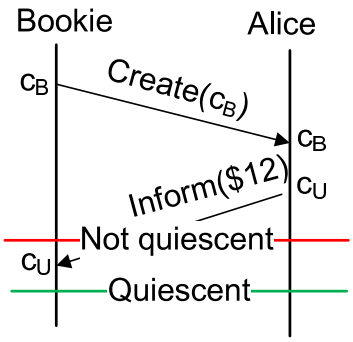

Figure 2.1: Quiescence

This means that there are no messages in transit in the system. Definition 3 states that an observation vector is *quiescent* if and only if every sent message has been received.

Definition 3 An observation vector $O \in \mathcal{O}_{\mathcal{A}}$ is quiescent iff $\forall x, y \in \mathcal{A}$, if $m(x, y)$ occurs in O_x, then $m(x, y)$ occurs in O_y.

2.3.2 Integrity

Consider Figure 2.2(A). Initially, Alice is committed to Bob that if the sky is clear, then she will meet him at the lake, meaning $c_L = \mathsf{C}(Alice, Bob, clear, lake)$. We model Bob's observation of the sky as a message that Bob receives from the environment Env. Now, Bob infers the unconditional commitment $c_{UL} = \mathsf{C}(Alice, Bob, \top, lake)$ whereas Alice does not yet infer c_{UL} (maybe because she is in a basement and cannot look at the sky). Thus, Bob and Alice would be misaligned. The main problem is that Bob has received some vital information that Alice might not have.

We wish to exclude observations by an agent where it has received vital information that it hasn't yet propagated to the other relevant parties. In Figure 2.2(B), it would be premature to consider alignment before Bob notifies Alice of *clear*. In this sense, Bob's notifying Alice of *clear* is integral with receiving $Inform(clear)$ from Env. Similarly, in Figure 3.3(B), Bookie sending the *Create* is integral with receiving $Inform(\$12)$. We recognize no intervening observations from the point of view of alignment until all integral observations have been made; in other words, the intervening observations are not visible. We now turn to the formalization.

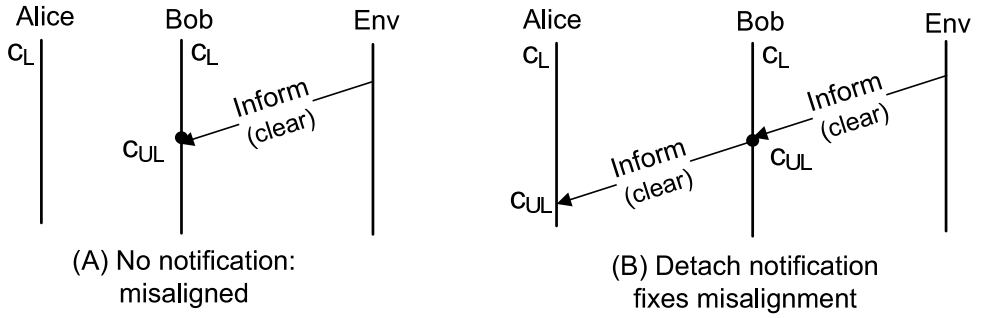

Figure 2.2: Notifying about detaches

We now show how to specify integrity constraints on observations. Chapter 3 specifies the integrity constraints relevant to alignment. First though, some preliminaries. Let O_x be an observation sequence of the form $\langle \ldots, m \rangle_x$. Then, for any message m', $O_x; m'$ is the concatenation of O_x with m', and is of the form $\langle \ldots, m, m' \rangle_x$. Let $\mathcal{S}(O_x)$ be the set of propositions that can be inferred from the observation sequence O_x. (Chapter 3 formalizes $\mathcal{S}(O_x)$.) If $p \notin \mathcal{S}(O_x)$, we say $\neg p \in \mathcal{S}(O_x)$. The empty condition ε is trivially in $\mathcal{S}(O_x)$. $\mathcal{S}(O_x)$ may be thought of as the *state* of x after observing the messages in O_x.

$\lfloor m[B : A]m' \rfloor_x$ is an *integrity constraint* on the observations of agent x. Here, B and A are the *before* and *after* conditions for the *trigger* m, and m' is the *effect* of m if the *before* and *after* conditions are met.

Definition 4 Consider a constraint $\lfloor m[B : A]m' \rfloor_x$. m' is an enabled effect of m with respect to an observation sequence o and the constraint iff $B \in \mathcal{S}(o)$ and $A \in \mathcal{S}(o; m)$.

An observation sequence O_x is *integral* with respect to a set of constraints iff for any prefix $o; m$ of O_x, $o; m; M$ is a prefix of O_x, where M *contains* an interleaving of the enabled effects of m with respect to o and the set of constraints.

An observation vector is *integral* with respect to a set of constraints iff each observation sequence in it is integral with respect to the set of constraints.

Definition 4 defines enabled messages as those that must be necessarily sent, as deduced from the integrity constraints. An observation sequence is not integral unless all enabled messages have been observed. Notice that to be integral, O_x must only *contain* the enabled effects (for every prefix); there is no restriction that the enabled effects must occur immediately after the trigger. This means that x may make extraneous observations

between the trigger and its enabled effects; however, those observations are not visible for the purposes of alignment.

2.3.3 Alignment

Now we formally define alignment—that agents would agree about whatever commitments as might result from any messages they might exchange. Definition 5 formalizes the notion of alignment by considering all potential observations of all agents.

Definition 5 A multiagent system $\mathcal{A}$ is *aligned* (written $[\![\langle\mathcal{A}\rangle]\!]$) iff $\forall O \in \mathcal{O}_{\mathcal{A}}$ such that O is quiescent and integral, $\forall x, y \in \mathcal{A} : \mathsf{C}(x, y, r, u) \in \mathcal{S}(O_y) \Rightarrow \mathsf{C}(x, y, r, u) \in \mathcal{S}(O_x)$.

Definition 5 considers the observations of creditors and debtors from the same integral and quiescent observation vectors. It says that if a creditor infers a commitment from its observations, then the debtor must infer that commitment from its own observations. When a debtor infers a commitment, but the creditor does not, no harm is done, and alignment is unaffected.

Chapter 3

Handling Autonomy and Distribution

Chapter 2 characterized commitment alignment. This chapter proposes the principles that *ensure* alignment despite distribution. In other words, if agents compute according to these principles, alignment is guaranteed. The principles essentially say how the commitments of agents should be computed, and how the agents must act to maintain alignment. We formalize the principles using the technical framework presented in Chapter 2, and prove that under the principles (and the communication model introduced in Chapter 2), any multiagent execution is aligned.

3.1 Introduction

$\mathsf{C}(\mathit{debtor}, \mathit{creditor}, \mathit{antecedent}, \mathit{consequent})$ means the debtor commits to the creditor that if antecedent holds, then the consequent will hold. An important insight in agent communication is that the interactions among agents may be understood in terms of their effects on the agents' commitments. For example, an offer for a copy of the book *Beating the Odds* from Bookie to Alice may be interpreted as $\mathsf{C}(\mathit{Bookie}, \mathit{Alice}, \$12, \mathit{BeatingtheOdds})$. In other words, Bookie commits that if Alice pays $12, then Bookie will deliver the book.

Imagine if Alice presumes that Bookie is committed to sending her the book she paid for, but Bookie is not committed to sending her the book. Their interaction would break down. In general, a key requirement for successful interaction is that the interacting

agents remain aligned with respect to their commitments. *Crucially, it turns out that even well-designed, well-behaved agents may become misaligned simply because of the distributed nature of the given system.* Previous approaches have largely ignored this problem or addressed it through restrictive, ad hoc assumptions. However, as commitment protocols expand into real-life distributed settings, a rigorous treatment becomes essential.

We consider realistic, distributed settings where agents communicate via asynchronous messaging. Asynchrony means that an agent is never blocked from sending a message. In such a system, the messages that the agents send each other may cross on the wire. Thus, in general, the agents may observe different messages in different orders. Since messages are understood in terms of their effects on commitments, the agents involved would become misaligned, i.e., come to conflicting conclusions about which commitments hold and which do not.

It is crucial to develop a formalization of commitments that ensures alignment despite asynchrony. First, distributed computing infrastructure is necessarily asynchronous. Large-scale systems exhibit high latency making synchronous interactions simply intractable in practice. Second, any formalization that works despite asynchrony also works in "more synchronous" settings, that is, those imposing additional constraints on agent behavior—for example, one where agents take turns sending messages. Third, asynchrony is inherently compatible with agent autonomy simply because an agent is never blocked from sending a message and, more pertinently, from acting upon its commitments.

In the absence of a formalization that supports reasoning about commitments in distributed settings, all research in applications of commitments is bound to report results that are either not general enough or are unduly complex. Such a formalization is currently missing; this paper seeks to fill this gap.

3.1.1 Motivation

Informally, we say that agents are aligned, if whenever an agent infers a commitment in which it is the creditor, the debtor of the commitment also infers that commitment. There are two possible causes of misalignment. One, the agents may assign incompatible meanings to the messages they are exchanging. Two, even when the agents assign identical meanings to the relevant messages, they may make incompatible observations. Chopra and Singh [Chopra and Singh, 2008] solve the former for a language similar to ours. This paper

addresses the second problem. Let's consider some examples to highlight the problem.

Example 7 (Figure 3.1(A)). Bookie sends Alice (a message that expresses) an offer that if she pays $12, then Bookie will deliver to her a copy of the book *Beating the Odds*. Alice sends Bookie a rejection of the offer. Upon receipt, Bookie resends the offer. ▮

As is typical in commitment protocols, Bookie's offer creates a commitment from Bookie to Alice for the book *Beating the Odds* in return for $12. In Example 7, both Alice and Bookie observe the messages in the same order, and therefore remain aligned.

Example 8 (Figure 3.1(B)). Bookie makes Alice an offer. Not seeing a response from Alice, Bookie resends the offer. Suppose that, in the meantime, Alice sends Bookie a rejection of the offer. Then the rejection crosses Bookie's repetition of the offer. ▮

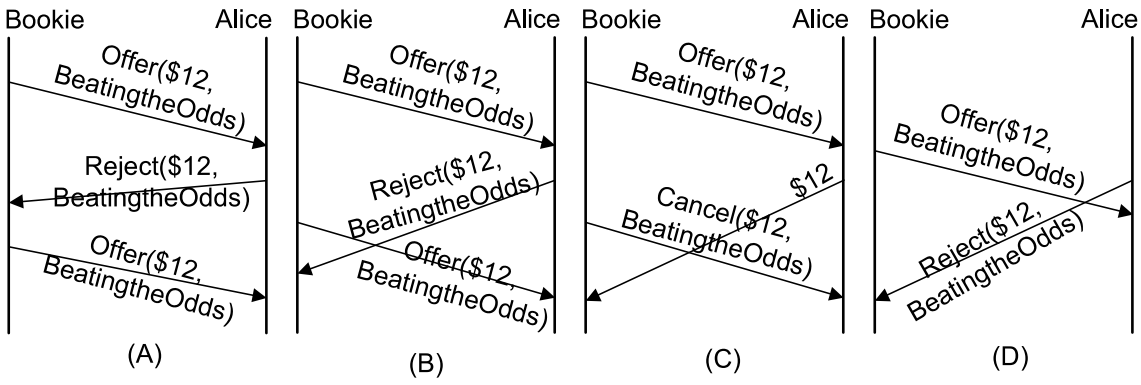

Figure 3.1: Scenarios (B),(C), and (D) end in misalignment

What ought Bookie and Alice to infer about the offer at the end of the exchange shown in Figure 3.1(B)? After seeing Alice's rejection of the offer, Bookie may infer that there no longer exists an offer to Alice. However, having seen an offer message last, Alice may infer that the offer holds. That is, Alice infers a commitment from Bookie for a copy of *Beating the Odds* for $12, whereas Bookie does not infer that commitment. This misalignment occurs because Alice's rejection and Bookie's offer messages crossed in transit. Note that Figures 7(A) and 7(B) imply a race condition between offer and rejection: their order (as viewed by Bookie) matters and yet Alice cannot distinguish between the two orders.

Example 9 (Figure 3.1(C)). Bookie makes an offer that Alice accepts and sends the payment for. In the meantime, Bookie cancels the offer. Bookie's cancellation and Alice's payment cross. ∎

In Example 9, upon sending the payment, Alice infers that Bookie is committed to sending her a copy of the book. Later, when Alice sees Bookie's cancel message, she regards it as spurious. However, Bookie sees the payment only after he has canceled its offer. So Bookie considers Alice's payment late. The result is that Alice infers an unconditional commitment for the book from Bookie, but one that Bookie does not infer. A race between cancellation and payment causes misalignment.

Example 10 (Figure 3.1(D)) Here, Bookie sends an offer, but in the meantime Alice sends a rejection. ∎

In the scenario in Example 10, Bookie infers the offer was rejected because that is the message it last sees, whereas Alice infers the offer exists because that is the message she last sees. Admittedly, the scenario is pathological: it makes no sense for Alice to reject an offer that Bookie never made. However, scenarios where messages arrive unexpectedly can occur when multiple parties are involved, and messages happen to be delayed differently on different paths. This is analogous to when one receives a group reply to an email before receiving the original email.

As the above examples demonstrate, asynchrony throws a major challenge in the face of alignment. Even agents who are perfectly designed and who assign identical meanings to messages may end up misaligned. Another way to cast this problem is in terms of the commitment operations, which show how to manipulate commitments [Singh, 1999]. Existing formalizations of the operations, e.g., [Desai et al., 2007b], do not support reasoning in distributed settings.

Current approaches for alignment fall into two main categories. Some use acknowledgements [McBurney and Parsons, 2003] as a way of serializing the operations in distributed settings. The idea is that the agents involved would observe the relevant messages in the same order, and hence make the same inferences. Such approaches are incompatible with autonomy. *Autonomy compatibility* means that no agent should have to wait for approval from other agents to effect a change in its commitments. In an acknowledgment-

based approach, for example, to effect a cancellation or discharge of a commitment, the debtor would have to seek the creditor's approval, which completely begs intuition.

Others suggest unique identifiers—there is no comprehensive proposal—for each commitment, and reference these identifiers from the update operations on commitments [Flores et al., 2004; Rovatsos, 2007]. Commitment identifiers fail to meet *semanticity*. Semanticity means that the proposal should accommodate general reasoning about commitments. For example, with identifiers, if $\mathsf{C}(id_0, x, y, r, u)$ and $\mathsf{C}(id_1, x, y, r, v)$ hold, semantically it still ought to be the case that $\mathsf{C}(\_, x, y, r, u \wedge v)$ holds ('_' is some identifier). To reason with identifiers, one would need to track dependencies for commitments a la distributed truth maintenance ([Huhns and Bridgeland, 1991]). Any such approach would be more complex than the approach presented here, without being more general.

3.1.2 Contributions

Our primary contribution is a formalization consisting of three elements: (1) messaging patterns that communicate the commitment operations; (2) a semantics of the operations that determines each participating agent's inferences regarding commitments; and (3) constraints on agent behavior described as messages the agents must send under specific circumstances. We prove that our formalization eliminates misalignments, and illustrate its intuitiveness and generality with the help of various examples.

Our contribution also lies in delineating two criteria that any approach for alignment must meet: *autonomy compatibility* and *semanticity*. Both autonomy compatibility and semanticity are crucial in arriving at a general solution for alignment. Our formalization meets both these criteria. In particular, our formalization does not rely upon using commitment identifiers as introduced above. However, its generality means that identifiers may be used as domain modeling artifacts if necessary.

3.1.3 Organization

The structure of this chapter is as follows. Section 3.2 introduces the principles of our approach. Section 3.3 formalizes the principles; Section 3.4 proves that alignment is guaranteed for all possible multiagent executions. Section 3.5 discusses related work and summarizes our contributions.

3.2 Principles of Alignment

The misalignments in Figure 3.1 are due to the naïve semantics that upon observing $\mathit{Create}(r, u)$, an agent infers $\mathsf{C}(r, u)$; upon observing $\mathit{Release}(r, u)$ or $\mathit{Cancel}(r, u)$ an agent infers $\neg\mathsf{C}(r, u)$.

We propose five principles that guarantee alignment. These principles are informed by the nature both of commitments and of distributed systems. Let us first consider three principles that address the misalignments in Figure 3.1.

Novel Creation. Observing $\mathit{Create}(r, u)$ should have no effect if a stronger commitment $\mathsf{C}(s, v)$ has held before.

Complete Erasure. Observing $\mathit{Release}(r, u)$ should have no effect if a *strictly* stronger commitment $\mathsf{C}(s, v)$ holds. If no such $\mathsf{C}(s, v)$ holds, then each weaker commitment $\mathsf{C}(r', u')$ is released. $\mathit{Cancel}(r, u)$ is analogous.

Accommodation. Observing $\mathit{Release}(r, u)$ has the effect that each weaker commitment $\mathsf{C}(s, v)$ is treated as if it has held before. $\mathit{Cancel}(r, u)$ is analogous.

Figure 3.2 shows how these principles restore alignment to the misaligned scenarios of Figure 3.1. Figure 3.2 shows *offer* as $\mathit{Create}(c_B)$, and *reject* as $\mathit{Release}(c_B)$.

Contrast Figures 3.1(A) and 3.2(A). In both figures, Bookie and Alice remain aligned at the end. However, in Figure 3.1(A), Bookie and Alice both infer c_B, whereas in Figure 3.2(A), neither of them infers c_B. Novel Creation supports Figure 3.2(A): the first offer causes c_B and resending the offer after receiving a reject has no effect.

Contrast Figures 3.1(B) and 3.2(B). In Figure 3.1(B), Alice infers c_B, whereas Bookie does not. In Figure 3.2(B), however, neither Alice nor Bookie infers c_B. Upon receiving the reject, because of Complete Erasure, Bookie considers itself released from the offer; receiving the same offer again has no effect on Alice because of Novel Creation.

Contrast Figures 3.1(D) and 3.2(C). In Figure 3.1(D), Alice infers c_B, whereas Bookie does not. In Figure 3.2(B), however, neither Alice nor Bookie infers c_B. Upon receiving the reject, because of Complete Erasure, Bookie considers itself released from the offer; receiving an offer which Alice has already rejected has no effect on Alice because of Accommodation and Novel Creation acting in concert. Accommodation ensures

that Alice's release of the offer makes it appear as if the offer had been made before, and hence when Bookie's actual offer arrives, NOVEL CREATION ensures the offer has no effect.

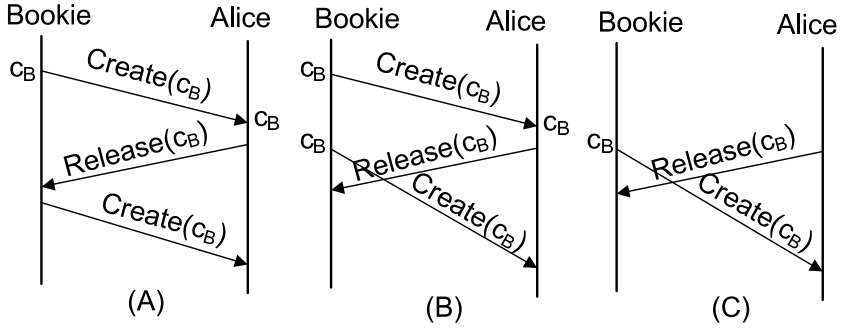

Figure 3.2: Proposed approach

NOVEL CREATION means that resending a *Create* of a previous commitment has no effect. In that case, how can Bookie again offer Alice essentially the same deal that she has rejected before? Circumstances might have changed, and Bookie might want to see if Alice will accept the offer this time around.

A possible domain modeling approach is to include identifiers on the conditions involved so as to distinguish the offers. In practice, we would place such identifiers anyway, so as to distinguish commitments made to different parties, e.g., to ensure that a different copy of the book would be delivered to each customer and each customer will pay for her purchase. Such identifiers do not apply on commitments and do not interfere with reasoning about commitments. In Example 11, at the end, both Alice and Bookie infer that the id_1 commitment holds and the id_0 commitment doesn't.

Example 11 Bookie sends $Create(Bookie, Alice, \$12(id_0), BeatingtheOdds(id_0))$. Alice sends $Release(Bookie, Alice, \$12(id_0), BeatingtheOdds(id_0))$. To offer the "same" deal again, Bookie sends $Create(Bookie, Alice, \$12(id_1), BeatingtheOdds(id_1))$. ▮

It is worth nothing that NOVEL CREATION does not say that if a commitment has held before, then it can never hold again; it only says that a *Create* message for such a commitment has no effect. A commitment may come to hold again because a *Create* message for a stronger commitment is observed. In real life, it is common practice for a seller to improve its offers, effectively making stronger commitments, as in Example 12.

Example 12 Bookie makes Alice the offer c_B. Alice rejects the offer thus releasing Bookie from c_B. However, Bookie is persistent, and it makes Alice the stronger offer c_0 (two books for the same price). This automatically resurrects c_B to ensure consistency. ▮

Example 13 Alice rejects Bookie's improved offer. ▮

When Alice sends $\mathit{Release}(c_0)$, Complete Erasure means that this not only removes c_0, but also c_B and c_G. Notice that partial releases are unsuccessful. Because c_0 is stronger than c_B, $\mathit{Release}(c_B)$ has no effect—c_0 continues to hold.

Notification. This principle ensures that two agent's states are compared only when both or neither has received vital information. This leads to two requirements. One, a creditor must notify relevant debtors of detaches, and a debtor must notify relevant creditors of discharges. Two, until an agent sends its pending notifications, it doesn't have a well-defined visible state. Reducing the visible states proves crucial because we can define alignment as agreement between the concerned agents at such states.

Figure 2.2(B) shows how alignment is preserved. The bold dot along Bob's lifeline indicates that Bob must send the *clear* notification to Alice. The middle state where Bob has detached the commitment but not notified Alice is excluded from consideration—it is not visible for the purposes of alignment. In this manner, we avoid a false negative claim about alignment. This case is of a creditor notifying the debtor of a detach. The case where a debtor notifies a creditor of a discharge is similar.

Priority. It is possible that a debtor cancels a commitment concurrently with the creditor detaching it.

Recall Example 9 where Alice's payment crosses Bookie's cancellation. Figure 3.3(A) annotates the same example with commitments. If Bookie's cancellation and Alice's payment cross, Alice and Bookie become misaligned—Alice infers $c_U = \mathsf{C}(\mathit{Bookie}, \mathit{Alice}, \top, \mathit{BeatingtheOdds})$ whereas Bookie does not. The reason is that receiving $\mathit{Cancel}(c_B)$ has no effect on Alice because she already infers c_U, which is a stronger commitment than c_B. Receiving Alice's \$12 payment has no effect on Bookie because there is no commitment to detach anymore. (The figures only show the strongest commitments. For example, because c_U entails c_B, Figure 3.3 never shows c_B in addition to c_U.)

There is no fundamental reason to prefer the creditor's or the debtor's viewpoint. For each commitment, the parties involved simply have to agree on what takes priority: cancel over detach, or detach over cancel. Detach priority means that the debtor considers its cancellation of a commitment to be overridden by the detach of the commitment. Cancel priority means that the creditor considers its detach of a commitment to be overridden by the debtor's cancellation of the commitment. Our theory handles both alternatives, and shows what the agents must do in each case. Consider a commitment $\mathsf{C}(r \wedge s, u)$. Suppose detach has priority over cancel. If the debtor observes a message that brings about s (a detach) after it has cancelled $\mathsf{C}(r \wedge s, u)$, then it must send $\mathit{Create}(r, u)$. Alternatively, suppose that cancel has priority over detach. If the creditor has already detached $\mathsf{C}(r \wedge s, u)$ by sending a message that brings about s, and it then observes a cancellation for $\mathsf{C}(r \wedge s, u)$, then the creditor must send $\mathit{Release}\mathsf{C}(r, u)$.

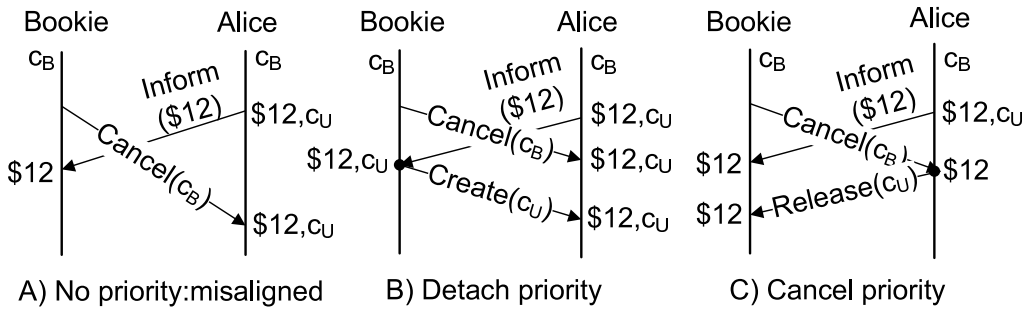

Figure 3.3: Race between cancel and detach

The protocol that Alice and Bookie are enacting would specify whether cancel or detach has priority for c_B. If detach has priority, then, as Figure 3.3(B) shows, Bookie considers its cancellation to be overridden by the detach, and creates c_U. If cancel has priority, then, as Figure 3.3(C) shows, Alice considers the detach to be overridden by the cancellation, and releases Bookie from c_U.

3.3 Formalization of the Principles

We introduce stable propositions $\mathit{created}(x, y, r, u)$, $\mathit{released}(x, y, r, u)$, and $\mathit{cancelled}(x, y, r, u)$, each corresponding to the eponymous commitment operation having occurred. Our formalization does not require propositions corresponding to the occurrence

of DELEGATE and ASSIGN. We adopt the postulates B9–B13 in addition to B1–B8.

B9. $\mathit{released}(r, u) \rightarrow \mathit{created}(r, u)$

B10. $\mathit{cancelled}(r, u) \rightarrow \mathit{created}(r, u)$

B11. $\mathit{created}(r, u)$ and $\mathsf{C}(r, u) \succeq \mathsf{C}(s, v) \Rightarrow \mathit{created}(s, v)$

B12. $\mathit{released}(r, u)$ and $\mathsf{C}(r, u) \succeq \mathsf{C}(s, v) \Rightarrow \mathit{released}(s, v)$

B13. $\mathit{cancelled}(r, u)$ and $\mathsf{C}(r, u) \succeq \mathsf{C}(s, v) \Rightarrow \mathit{cancelled}(s, v)$

Let's consider some examples to see how B9–B13 work. Suppose $\mathit{created}(c_0)$ holds; by B11, $\mathit{created}(c_B)$ and $\mathit{created}(c_G)$ hold. Suppose $\mathit{released}(c_0)$ holds; by B9, $\mathit{created}(c_0)$ holds too; by B12, $\mathit{released}(c_B)$ and $\mathit{released}(c_G)$ hold; by B9, $\mathit{created}(c_B)$ and $\mathit{created}(c_G)$ hold. Cancelled commitments are treated as analogous to released commitments.

Let's see how B9–B13 relate to the principles introduced earlier. B12 and B13 relate to COMPLETE ERASURE. If a commitment is released or if it is cancelled, all weaker commitments are released or cancelled, as may be the case. B9 and B10 (together with B12 and B13) portray ACCOMMODATION: if a commitment has been cancelled or released, treat all weaker commitments as if they had held. B11 relates to NOVEL CREATION. It ensures that once $\mathit{created}(r, u)$ holds, all commitments weaker than $\mathsf{C}(r, u)$ are also considered created.

Now we define the semantics of the operations themselves in terms of $\mathcal{S}(o)$, the set of propositions that can be inferred from the observation sequence o. For any set of propositions $\mathcal{Q}$, $\mathcal{Q}^*$ is the deductive closure of $\mathcal{Q}$. $\mathcal{Q}^\Pi$ is the *atomic projection* of $\mathcal{Q}$ such that a proposition q belongs to $\mathcal{Q}^\Pi$ if and only if two conditions are satisfied: (1) q belongs to $\mathcal{Q}$, and (2) q is either an atomic proposition, or of the form $\mathsf{C}(r, u)$, or of the form $\mathit{created}(r, u)$ or $\mathit{released}(r, u)$ or $\mathit{cancelled}(r, u)$.

Let $\mathcal{S}(o_x)$ be the current state of x. The general pattern for computing the state $\mathcal{S}(o_x; m)$ is the following. First modify $\mathcal{S}(o_x)$ by adding or removing propositions relevant to m. Let $\mathcal{S}'(o_x; m)$ be the resulting set . $\mathcal{S}(o_x; m)$ is $(\mathcal{S}'(o_x; m)^*)^\Pi$, in other words, the atomic projection of the deductive closure of $\mathcal{S}'(o_x; m)$. Let us facilitate this pattern by introducing the notation $\mathcal{Q}^\odot$, the *atomic closure* of $\mathcal{Q}$, to mean $(Q^*)^\Pi$.

3.3.1 Inform

B14 is the semantics of $\mathit{Inform}(r)$: r holds upon observing it.

B14. $\mathcal{S}(o; \mathit{Inform}(r)) = (\mathcal{S}(o) \cup \{r\})^{\odot}$

3.3.2 Two-Party Operations

The messages $\mathit{Create}(r, u)$, $\mathit{Release}(r, u)$, and $\mathit{Cancel}(r, u)$ realize the corresponding operations.

B15 and B16 give the semantics of $\mathit{Create}(r, u)$. B15 states that if $\mathit{created}(r, u)$ or the consequent u already hold, then upon observing $\mathit{Create}(r, u)$, we insert $\mathit{created}(r, u)$, and compute its atomic closure to obtain the resulting state. In particular, $\mathsf{C}(r, u)$ does not hold in the resulting state. The condition related to consequent u is present because the consequent of the commitment and the commitment both holding together is inconsistent according to B1. Hence, if u holds, $\mathit{Create}(r, u)$ has no effect. Conversely, B16 states that if neither $\mathit{created}(r, u)$ nor u holds in the current state, then upon observing $\mathit{Create}(r, u)$, we insert $\mathsf{C}(r, u)$ and $\mathit{created}(r, u)$, and compute the atomic closure to obtain the resulting state.

B15. $\mathit{created}(r, u) \in \mathcal{S}(o)$ or $u \in \mathcal{S}(o) \Rightarrow \mathcal{S}(o; \mathit{Create}(r, u)) = (\mathcal{S}(o) \cup \{\mathit{created}(r, u)\})^{\odot}$

B16. $\mathit{created}(r, u) \notin \mathcal{S}(o)$ and $u \notin \mathcal{S}(o) \Rightarrow \mathcal{S}(o; \mathit{Create}(r, u)) =$
$(\mathcal{S}(o) \cup \{\mathsf{C}(r, u), \mathit{created}(r, u)\})^{\odot}$

Let $\lceil\lceil \mathsf{C}(r, u) \rceil\rceil$ denote the set $\{\mathsf{C}(s, v) | \mathsf{C}(r, u) \succeq \mathsf{C}(s, v)\}$, that is, the set of commitments weaker than $\mathsf{C}(r, u)$. According to B17, upon observing $\mathit{Release}(r, u)$, we remove all commitments weaker than $\mathsf{C}(r, u)$, insert $\mathit{released}(r, u)$, and then compute the atomic closure to obtain the resulting state. B18 analogously gives the semantics of $\mathit{Cancel}(r, u)$.

B17. $\mathcal{S}(o; \mathit{Release}(r, u)) = ((\mathcal{S}(o) \setminus \lceil\lceil \mathsf{C}(r, u) \rceil\rceil) \cup \{\mathit{released}(r, u)\})^{\odot}$

B18. $\mathcal{S}(o; \mathit{Cancel}(r, u)) = ((\mathcal{S}(o) \setminus \lceil\lceil \mathsf{C}(r, u) \rceil\rceil) \cup \{\mathit{cancelled}(r, u)\})^{\odot}$

B9–B18 accurately capture Novel Creation, Complete Erasure, and Accommodation.

3.3.3 Three-Party Operations

Clearly, any implementation of DELEGATE and ASSIGN must involve at least two messages. Figure 3.4(A) exemplifies the message pattern for delegation. Bookie (the delegator) delegates c_B to Charlie (the delegatee). Bookie sends $Delegate(c_B, Charlie)$ to Charlie. Let $d\_c_B = \mathsf{C}(Charlie, Alice, \$12, BeatingtheOdds)$. Upon its receipt, Charlie sends $Create(d\_c_B)$ to Alice, thus fully realizing the delegation. Figure 3.4(B) exemplifies the message pattern for assignment. Here, Alice (the assignor) wants to assign c_B from Bookie to Bob (the assignee). Alice sends $Assign(c_B, Bob)$ to Bookie. Let $a\_c_B = \mathsf{C}(Bookie, Bob, \$12, BeatingtheOdds)$. Upon its receipt, Bookie sends $Create(a\_c_B)$ to Bob, thus fully realizing the assignment.

B19 and B20 state the semantics of *Delegate* and *Assign* messages, respectively: observing either of these messages has no direct effect on the agent.

B19. $\mathcal{S}(o; Delegate(x, y, z, r, u)) = \mathcal{S}(o)$

B20. $\mathcal{S}(o; Assign(x, y, z, r, u)) = \mathcal{S}(o)$

The computation of $\mathcal{S}(o)$ is closed under B14–B20.

In the delegate and assign patterns, the initiating messages—*Delegate* and *Assign*, respectively—are *instructions* to an agent to create a new commitment. R1 and R2 in Table 3.1 capture the instructional nature of the delegate and assign messages, respectively, as integrity constraints. Each row in Table 3.1 is in fact, a constraint on agent behavior, and is of the form $\lfloor Trigger[Before : After]Effect \rfloor_{Agent}$. For example, R1 is $\lfloor Delegate(x, y, z, r, u)[\varepsilon : \varepsilon]Create(z, y, r, u) \rfloor_z$. R3–R8 are explained below.

There are a few points of note about delegation and assignment as presented here.

- R1 and R2 have nothing to do with restoring alignment. That the *Create* must follow the instruction simply alludes to the atomicity of delegation and assignment as operations.

- Delegation does not involve a notification from the delegator to the creditor that the commitment is being delegated. No doubt, such notifications could be practically valuable; however, our aim here is to delineate the core patterns on top of which additional patterns, such as those involving a notification to the creditor may be built.

For the same reason, assignment does not involve a notification from the assignor to the assignee.

- The new commitment must be explicitly created by the debtor—the delegatee in the case of delegation and the debtor in the case of assignment. This reflects upon a principled approach for manipulating commitments, by reusing the semantics of *Create* above.

- If Bookie delegates c_B twice to Charlie, then the second time Charlie need not send a *Create*: such a message would be useless under Novel Creation. This paper sacrifices optimization in favor of simplicity.

Considerations of when a commitment operation may successfully occur are beyond our scope (for delegation, [Norman and Reed, 2001] offers an interesting discussion). This papers assumes that all operations are successful. Hence, even though Figure 3.4(A) shows c_B to hold before delegation is initiated, that should not be interpreted as a success precondition for delegation. Even if Bookie did not infer c_B initially, Bookie's delegate message to Charlie would still cause Charlie to send the create message to Alice.

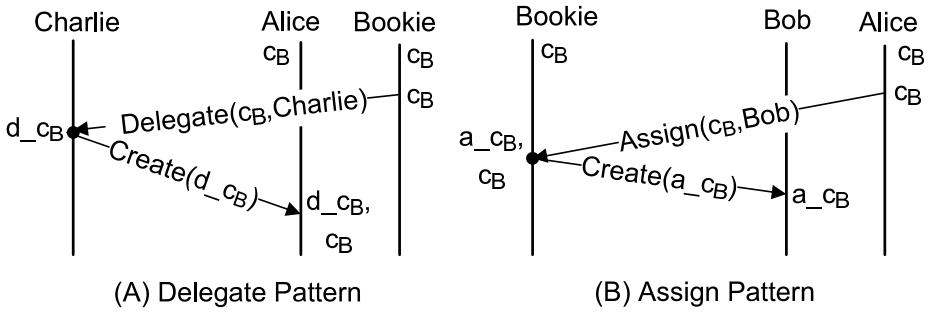

Figure 3.4: The delegate and assign patterns

Table 3.1: Constraints on agent behavior

#	**Name**	**Agent**	**Trigger**	**Before**	**After**	**Effect**
R1	Delegate	z	$Delegate(x, y, z, r, u)$	ε	ε	$Create(z, y, r, u)$
R2	Assign	x	$Assign(x, y, z, r, u)$	ε	ε	$Create(x, z, r, u)$
R3	Detach1	y	$Inform(z, y, s)$	$\mathsf{C}(x, y, r \wedge s, u) \wedge \neg\mathsf{C}(x, y, r, u) \wedge \neg s$	s	$Inform(y, x, s)$
R4	Detach2	y	$Create(x, y, r \wedge s, u)$	$\neg\mathsf{C}(x, y, r \wedge s, u) \wedge s$	$\mathsf{C}(x, y, r \wedge s, u)$	$Inform(y, x, s)$
R5	Discharge1	x	$Inform(z, x, u)$	$\mathsf{C}(x, y, r, u) \wedge \neg u$	u	$Inform(x, y, u)$
R6	Discharge2	x	$Create(x, y, r, u)$	$\neg\mathsf{C}(x, y, r, u) \wedge u'$ where $u \vdash u'$	ε	$Inform(x, y, u')$
R7	D-Priority	x	$Inform(z, x, s)$	$cancelled(x, y, r \wedge s, u) \wedge \neg\mathsf{C}(x, y, r \wedge s, u) \wedge \neg s$	s	$Create(x, y, r, u)$
R8	C-Priority	y	$Cancel(x, y, r \wedge s, u)$	$s \wedge \mathsf{C}(x, y, r \wedge s, u) \wedge \neg\mathsf{C}(x, y, r', u')$ such that $\mathsf{C}(x, y, r', u') \succ \mathsf{C}(x, y, r, u)$	ε	$Release(x, y, r, u)$

3.3.4 Notifications

Recall that NOTIFICATION states that creditors must notify debtors of detaches, and debtors must notify creditors of discharges. Two cases arise for each kind.

Detach1 (R3). y infers $\mathsf{C}(x, y, r \wedge s, u)$ and $\neg\mathsf{C}(x, y, r, u) \wedge \neg s$, meaning that the commitment is not detached yet. y then observes $\mathit{Inform}(s)$ from some z. As a result, $\mathsf{C}(x, y, r \wedge s, u)$ is detached, and y infers $\mathsf{C}(x, y, r, u)$. y must now inform x about the detach by sending $\mathit{Inform}(y, x, s)$.

Detach2 (R4). y infers s and $\neg\mathsf{C}(x, y, r \wedge s, u)$, and then observes $\mathit{Create}(x, y, r \wedge s, u)$. Therefore, y infers $\mathsf{C}(x, y, r \wedge s, u)$. $\mathsf{C}(x, y, r \wedge s, u)$ is detached upon s; hence, y infers $\mathsf{C}(x, y, r, u)$. y must now inform x about the detach by sending $\mathit{Inform}(y, x, s)$.

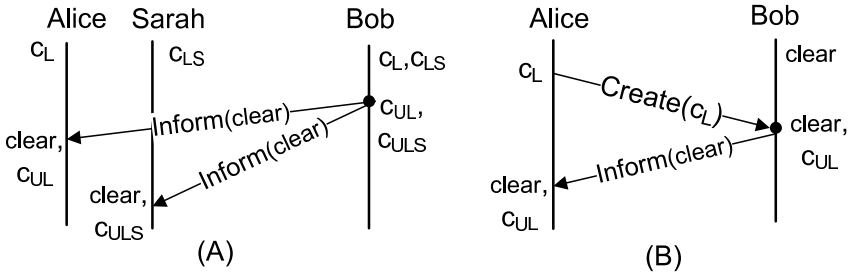

Figure 3.5: Detach notifications

Figure 2.2(B) illustrates R3. When Bob receives $\mathit{Inform}(\mathit{clear})$, R3 kicks in and ensures Alice is notified, thus preserving alignment. Figure 3.5(A) is another example of R3 at work. Here Alice and Sarah are committed to meeting Bob at the lake if the sky is clear (c_L and c_{LS}, respectively). At some point, Bob figures the sky is clear and therefore infers that both Alice and Sarah are now unconditionally committed to meet him (c_{UL} and c_{ULS}, respectively). R3 ensures that both Alice and Sarah are notified that the clear condition has been met, thus preserving alignment. Figures 3.5(B) illustrates R4. Here, Bob already infers *clear*. So when Bob receives $\mathit{Create}(c_L)$, Bob infers that Alice is unconditionally committed (c_{UL}). R4 kicks in and ensures Alice is notified.

Discharge1 (R5). x infers $\mathsf{C}(x, y, r, u)$ and $\neg u$. x then observes $\mathit{Inform}(u)$ from some z. As a result, $\mathsf{C}(x, y, r, u)$ is discharged. x must now inform the creditor y of the discharge by sending $\mathit{Inform}(x, y, u)$.

Discharge2 (R6). x infers u'. x then sends $Create(x, y, r, u)$ such that $u \vdash u'$. x will not infer $\mathsf{C}(x, y, r, u')$ because u' holds. However, y may not yet infer u'. Therefore, y may infer $\mathsf{C}(x, y, r, u')$. Hence, x must now send $Inform(x, y, u')$.

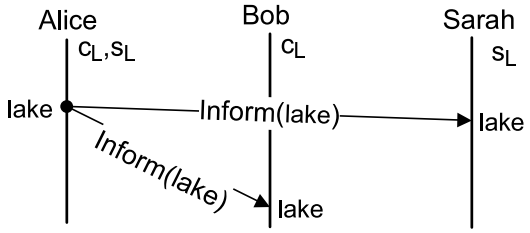

Figure 3.6: Discharge notification

Figure 3.6 illustrates the usage of R5. Alice is committed to both Bob and Sarah to be at the lake (c_L and s_L, respectively). When Alice gets to the lake, she discharges those commitments. R5 kicks in and ensures that both Bob and Sarah are informed accordingly so that they also consider their respective commitments discharged.

In Figure 3.4(A), after Alice observes the create message from Charlie, suppose Alice sends Bookie $Inform(\$12)$ (if she already inferred \$12, then upon observing the create, R4 would apply). This detaches c_B. Then R3 kicks in and ensures that Alice also sends Charlie $Inform(\$12)$. This should not be taken to mean that Alice sends \$12 each to Bookie and Charlie—the proposition \$12 is semantically no different than the proposition *clear*. An analogous argument can be made for the scenario in Figure 3.4(B). Suppose that after Bookie sends the create message, it sends $Inform(BeatingtheOdds)$ to Alice. Now R5 would ensure that Bookie also sent $Inform(BeatingtheOdds)$ to Bob.

3.3.5 Priority

Below, we formalize the implications of detach priority and cancel priority for a commitment $\mathsf{C}(x, y, r \wedge s, u)$.

Detach Priority (R7). x infers $cancelled(x, y, r \wedge s, u)$ and $\neg\mathsf{C}(x, y, r \wedge s, u) \wedge \neg s$. (Note that $cancelled(x, y, r \wedge s, u) \not\Rightarrow \neg\mathsf{C}(x, y, r \wedge s, u)$. A cancelled commitment may come to hold again because a stronger commitment was created.) x then observes $Inform(s)$ from some z. If $\mathsf{C}(x, y, r \wedge s, u)$ had not been cancelled, it would have been detached. But y may

not know about the cancellation yet. Therefore, the debtor must act as if the commitment has been detached. Hence, it must now send $\mathit{Create}(x, y, r, u)$.

Cancel Priority (R8). y infers s and $\mathsf{C}(x, y, r \wedge s, u)$. Therefore, it also infers $\mathsf{C}(x, y, r, u)$. y then observes $\mathit{Cancel}(x, y, r \wedge s, u)$. It could be that x sent $\mathit{Cancel}(x, y, r \wedge s, u)$ without knowing that s holds, and therefore x may not infer $\mathsf{C}(x, y, r, u)$. To fix this possible misalignment, y must now send $\mathit{Release}(x, y, r, u)$. y though does not have to send the release if a commitment strictly stronger than $\mathsf{C}(x, y, r, u)$ holds. Sending the release then will be ineffective because of Complete Erasure.

Figure 3.3(B) illustrates the case of detach priority to fix the misalignment of Figure 3.3(A), whereas Figure 3.3(C) illustrates the case of cancel priority.

It could be that in the case of detach priority, Alice cheats by sending the payment even after receiving the cancel. Analogously, in the case of cancel priority, Bookie could cheat and get away with it. In settings where the parties are mutually untrustworthy, we can imagine the use of techniques such as secure mediators to ensure that neither party deceives the other.

R1–R8 are weak and locally executable constraints on an agent's behavior because they only call for an agent to *send* messages. They involve neither receiving a message nor synchronizing with another agent.

3.4 Correctness Proof

Now it remains to show that under the assumptions we have made, the formalization of commitment operations we have proposed guarantees that any multiagent system is aligned. Notice that a commitment is strengthened only through a *Create* or an *Inform* (as detach). A commitment is removed or weakened only through a *Release* or *Cancel*, or an *Inform* (as discharge).

Theorem 1 For any $\mathcal{A}$, A1–A4, B1–B20 and R1–R8 guarantee alignment, that is, $[\![\mathcal{A}]\!]$. ∎

Proof. $\mathcal{A}$ is aligned at the outset, i.e., in the observation vector of empty sequences, when no agent has made any observations. Inductively, assume that $\mathcal{A}$ is aligned up to a quiescent, integral observation vector O. Consider two agents, x and y in $\mathcal{A}$.

Now expand O to a quiescent, integral observation vector $O' = O; O^{\Delta}$. There are two possible threats to alignment: (1) if y infers a new commitment as creditor that its

debtor doesn't; and (2) if y continues to infer a commitment as creditor that it previously inferred, but its debtor no longer does.

For (1), consider a commitment added by y, i.e., $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O'_y) \setminus \mathcal{S}(O_y)$. Without loss of generality, assume $\mathsf{C}(x, y, r, u)$ is maximally strong, i.e., no other commitment added by y is strictly stronger than $\mathsf{C}(x, y, r, u)$. This means O_y^{Δ} includes receiving a detach (*Inform*) or a create message. For a detach, by integrity, y would have sent a message to x, which would have landed within O_x^{Δ} to ensure quiescence. A create would have originated from x. In either case, the quiescence of O' ensures that $O'_x \vdash \mathsf{C}(x, y, r, u)$.

For (2), consider a commitment not added by y but removed by x, i.e., $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O_y)$ and $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O_x) \setminus \mathcal{S}(O'_x)$. Without loss of generality, assume $\mathsf{C}(x, y, r, u)$ is maximally strong, i.e., no other commitment removed by x is strictly stronger than $\mathsf{C}(x, y, r, u)$.

Because $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O_y)$, by our inductive hypothesis, $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O_x)$. Hence, if $\mathsf{C}(x, y, r, u) \notin \mathcal{S}(O'_x)$, this means O_x^{Δ} includes receiving a discharge or release, or sending a cancel message. The release would be sent by y, thus $\mathsf{C}(x, y, r, u) \notin \mathcal{S}(O'_y)$. The cancel would be sent to y and the discharge would be propagated to y to ensure integrity. Therefore, by quiescence, $\mathsf{C}(x, y, r, u) \notin \mathcal{S}(O'_y)$.

3.5 Discussion

We delineated the problem of commitment alignment in multiagent systems and proposed a solution to it. Our solution consists of three parts: (1) messaging patterns to implement commitment operations in a distributed setting, (2) a semantics of the commitment operations, and (3) local constraints on agent behavior. We have adopted a general model of multiagent systems where communication is asynchronous but reliable. Our formalization of the commitment operations meets both autonomy compatibility and semanticity.

3.5.1 Generality of Approach

Although alignment is recognized as a problem in implementing agents, there exist no satisfactory proposals that guarantee alignment in a truly asynchronous setting. Most proposals either use a shared commitment store [McBurney and Parsons, 2003] or revert to an approach based on commitment identifiers wherein every created commitment

is given a new identifier and update operations on commitments reference those identifiers [Flores et al., 2004; Rovatsos, 2007]. We reject the use of commitment identifiers because identifiers are a syntactic embellishment and, for the sake of generality, we are interested in a purely semantic solution. To illustrate our point, even if we used identifiers, if $\mathsf{C}(id_0, x, y, r, u)$ and $\mathsf{C}(id_1, x, y, r, v)$ hold, semantically it still ought to be the case that $\mathsf{C}(\_, x, y, r, u \wedge v)$ holds ('_' means some identifier). In the interest of obtaining a purely semantic solution, we advocate that if identifiers have to be used, then they should be used as a parameters of the propositional content of commitments as we did in Example 11, where we used timestamps as identifiers. Reasoning about identifiers may be needed in some cases, for example, when a merchant combines two orders into one because they are shipping to the same party and the same address [Wang and Miller, 2005]. Such specialized reasoning can be built on top our scheme.

The delegate and assign operations deserve special mention. Even if the delegatee ignores the delegate message, there won't be a misalignment (see B19). Hence, it would appear that there is no need to impose a restriction that upon receiving the message the delegatee *must* send a create message. By contrast, we conceptualize delegation as an indivisible operation, and we assume that the necessary conditions exist for *successful* delegation [Castelfranchi, 1998; Norman and Reed, 2001]. An analogous argument can be made for assign. Norman and Reed, in particular, distinguish between responsibility for bringing about a state of affairs versus responsibility for doing an action, and relate the distinction to that between extensional and whole-hearted satisfaction [Norman and Reed, 2001]. Commitments are tied to responsibility in that, broadly speaking, an agent is responsible for that to which it is committed. However, unlike Norman and Reed, we only model commitments to states of affairs, not to actions.

Winikoff [2007] studies how commitments may be implemented in a distributed setting. However, his solution only allows for a monotonically increasing set of commitments, and does not support discharge, release, and cancel.

McBurney and Parsons [2003] criticize the commitment operations as proposed by Singh (and as we have adopted here) on grounds that in e-commerce settings, a party should not be able to unilaterally manipulate a commitment. For example, they find it problematic that a debtor could unilaterally cancel a commitment without agreement from the creditor. But that would mean a debtor may never cancel a commitment because the creditor needs

to approve the cancel later. Similarly, McBurney and Parsons consider a commitment to be discharged only when both creditor and debtor agree that it is discharged. Again, a creditor may never accept that a commitment is discharged. In general, the assumption that all parties must agree on something for it to have effect is autonomy-incompatible, and does not reflect real-world scenarios. This is borne out by numerous clauses in the Uniform Commercial Code article concerning sales [UCC-Sales]. It almost seems that McBurney and Parsons are arguing backwards: if the parties are aligned after the manipulation of some commitment, only then the commitment should be considered manipulated.

Our formalization of the commitment operation identifies the fundamental multiparty messaging patterns. Other business patterns could be built on top. For example, the delegation pattern we have presented may be thought of as *delegation while retaining responsibility* since the delegator remains committed too. A *delegation without responsibility* pattern would additionally involve a cancellation message from the delegator to the creditor. Another example is division of labor. Singh *et al.* describe several such patterns from an architectural point of view [2008].

3.5.2 Applications

Our approach can benefit areas where commitments are used as the central basis for semantics. The connection with communication languages [Flores et al., 2004], [Fornara and Colombetti, 2004] and protocols [Desai et al., 2007b] is the most obvious.

Commitments are central to argumentation [Bentahar et al., 2004; Amgoud et al., 2002; Prakken, 2005]. Players in a dialogue game are envisaged as having private commitment stores. In most current work, a dialogue protocol, which limits how and when the players may make moves, also helps to keep the agents aligned. However, dialogue protocols may be unduly restrictive; for example, it may only allow turn taking. Our results could lead to more flexible and robust dialogue protocols.

Several researchers [Winikoff, 2007; Bordini et al., 2007] see commitments as an important high-level abstraction for Agent-Oriented Programming Languages. The key obstacle to providing commitments as an abstraction was the lack of a formal semantics of commitments. Now with a semantics suitable for distributed systems, it should be possible to make progress in that direction.

3.5.3 Multiagent Belief Consistency

In general, the problem of commitment alignment may be compared with the problem of belief *consistency* in multiagent systems. Commitment and beliefs are fundamentally different in that commitments are public artifacts whereas beliefs are private. Hence, maintaining belief consistency has largely meant that each agent maintains a locally consistent belief set. The notion of *shared* beliefs does have some utility in situations where agents are cooperative and need to coordinate their actions based on their shared beliefs. Then it becomes important that agents are globally consistent in their shared beliefs [Huhns and Bridgeland, 1991]. However, assuming that agents are cooperative is a limiting assumption; it does not hold for open systems where agents are autonomous and heterogeneous. For open systems, reasoning about commitment alignment is more relevant than reasoning about shared beliefs [Singh, 1998].

There is another important distinction between commitment alignment and belief consistency. Belief consistency is symmetric: if x believes p, and p is a belief shared with y, then for the system to be consistent, y must believe p too. However, commitment alignment is asymmetric: if the debtor infers a commitment that the creditor does not, it is not a misalignment. One may argue for a stricter definition of commitment alignment where both agents infer exactly the same set of commitments. However, given that alignment also serves as the basis for commitment-level interoperability [Chopra and Singh, 2008], the stricter definition would be *far too strict*. For instance, if Bookie takes the offer to mean $\mathsf{C}(Bookie, Alice, \$12, copy(BeatingTheOdds) \wedge copy(GamblingTips))$ whereas Alice takes it to mean $\mathsf{C}(Bookie, Alice, \$12, copy(BeatingTheOdds))$, we would like to say that Bookie and Alice are interoperable.

Paurobally *et al.* [2003] are motivated by the same challenges as are addressed in this paper. Their solution to alignment, however, is different from ours in at least two significant ways. One, they consider the consistency of beliefs across agents, in other words, belief alignment. Two, they propose synchronization protocols that keep the agents aligned. They consider two asynchronous communication models: reliable, which is similar to our model, and unreliable. The synchronization protocol for the reliable model is an acknowledgement-based commitment alignment scheme, which we rejected. As for the model where communication is unreliable, we did not consider it given the ubiquity of

reliable messaging middleware. Moreover, the synchronization protocols proposed by Paurobally *et al.* result in the agents behaving in a lock-step fashion. In contrast, our agents interact asynchronously.

3.5.4 Service-Oriented Architectures

Alignment mechanisms for services mirror those developed for distributed database systems, wherein the mechanisms typically involves of a commit protocol that serves to synchronize the participants. WS-Coordination is a generic framework within which such commit protocols may be defined [OASIS, 2007]. The use of commit protocols is closely tied with viewing business processes as workflows [Curbera et al., 2003]. Instead, we understand business processes in terms of the intrinsic meaning of the messages exchanged.

The trend lately is to use *business protocols* as modeling abstractions for business processes as evidenced by the growing number of such specifications [RosettaNet, 1998; TWIST; HL7]. General purpose frameworks around business protocols are also emerging [WS-CDL, 2005; ebBP, 2006]. State alignment remains a critical challenge for these frameworks. The move toward protocols is a positive step. Unfortunately, when it comes to state alignment, the tendency remains to fall back upon synchronization protocols. The reason this is so is that these frameworks ignore the business meanings of messages.

Molina-Jimenez *et al.* pose an interesting alignment problem in business conversations [2007]. It could happen that a participant discards a received message because the message fails to meet some arbitrary application-specific semantic constraint. For example, if a revenue officer finds Alice's tax filing to be invalid because she filled out the wrong form, the officer infers that Alice has yet to discharge her commitment to file taxes. Such scenarios would ideally be captured in our approach via compensation. Let's say $\mathsf{C}(\mathit{Alice}, \mathit{Officer}, \top, \mathit{fileTaxes}(\mathit{id}_0))$, that is, Alice commits to the revenue officer to file taxes. Upon receiving the invalid form, the revenue officer considers Alice's commitment discharged. However, the officer sends Alice a message indicating that the form was invalid, which detaches the commitment $\mathsf{C}(\mathit{Alice}, \mathit{Officer}, \mathit{invalid}, \mathit{created}(\mathit{Alice}, \mathit{Officer}, \top, \mathit{fileTaxes}(\mathit{id}_1)))$.

Chapter 4

Handling Heterogeneity

The formalization in Chapter 3 guarantees that agents will not be misaligned because of concerns arising from the nature of distributed systems. The formalization is really *middleware*: it works behind the scenes. Agent designers don't have to be aware of it. This chapter considers the interfaces of agents, that is, what individual messages count as for specific agents. It presents a simple interface specification language, and presents a decision procedure that checks if interfaces of agents are compatible with respect to commitments. Before agents engage each other, it is necessary to run this procedure to make sure that no alignment will arise because of differing interpretation of messages.

4.1 Introduction

Interoperability is a matter of *manifest agreement*. In other words, the interoperability of two or more parties means not only that there is an agreement among the parties but also that they can act according to the agreement. An *agent* is a computational representation of a "real" business principal. Agents interact with each other and their environment. We restrict attention to arms-length interactions in the form of *communications* among agents. These may be naturally realized in the computational infrastructure through messaging. For concreteness, we refer to the elements of communication as messages.

The scope of the agreement among the agents determines the scope of their interoperability. Communicating agents may thus interoperate at the level of text encoding (as in ASCII), syntax (as in XML InfoSet), grammar (as in UBL, the universal business

language or more simply the specification of a purchase order), messaging (as in SOAP), terminology (as in the Dublin Core vocabulary), and so on. Effective interoperation among two or more agents presumes that they are interoperable at all relevant levels.

As agents communicate, they enter into *commitments* with one another. The commitments reflect the organizational or social relationships among the agents, and thus characterize their interactions at a high level. We propose a commitment-based theory of interoperability. This approach reflects the intuition that the most relevant—and least implementation dependent—kind of agreement is based on the commitments that the agents have toward one another. Thus, agents are deemed interoperable if they can enter into and maintain well-aligned commitments to each other. Commitments represent an essential level at which to assess and establish interoperability because they yield a notion of compliance eminently suitable for open settings: the principals may act as they please provided it is in accordance with their commitments. Ensuring or verifying that agents act according to their commitments is a different challenge [Venkatraman and Singh, 1999].

Early in the study of software architecture, Parnas proposed that connectors between components should be treated not as control or data flow constructs but as *assumptions* made by each component about the others [Parnas, 1971]. Arguably, much of the subsequent work on software architecture and interoperability regressed from Parnas' insight: it has primarily considered connectors and concomitant assumptions at the level of flow, e.g., dealing with message order and occurrence [Hohpe and Woolf, 2004]. Such low-level criteria are largely orthogonal to considerations of business meaning. It is generally irrelevant whether the parties communicate via a procedure call or a message, and whether they follow a specific message order (unless the message order has a bearing on the meaning). Specifically, just because two agents are able to interact according to a specified *choreography* (i.e., a description of message ordering and occurrence) doesn't mean that their principals agree on the business meaning of the messages they exchange. Thus existing and emerging standards such as the Web Services Choreography Description Language (WS-CDL) [2005] apply at too low a level of abstraction.

By contrast, commitments enable naturally expressing the assumptions that agents make of other agents regarding the business meanings of their interactions. What matters at the business level is what commitments exist, not what low-level means are used to create or manipulate a commitment. Of course, checking commitment-level interoperability does

not obviate the need for checking the other kinds of interoperability, such as those alluded to above. But checking other kinds of interoperability is rarely adequate, and we need ways to define and check commitment-level interoperability, which is what this paper seeks to do.

4.1.1 Commitments

Commitments help us address business level interoperability.

For us, interoperability is concerned with whether the agents involved can enter into and maintain well-aligned commitments with each other. Stated informally, this means that if an agent's state models a commitment of which the agent is the creditor, then the debtor's state must also model the same commitment. In other words, the debtor *covers* the creditor's assumption about the commitment. For example, let's say a customer takes a quote message to mean that the merchant commits to sending goods if the customer pays first, whereas the merchant takes it to mean no such commitment. This problem can arise in foreign exchange transactions as well [Desai et al., 2007a]. The above illustrates commitment misalignment: on receiving the message, the customer's state models a commitment in which it is the creditor and the merchant the debtor, but the merchant's state does not reflect this commitment. They are thus noninteroperable.

The reverse condition—if a debtor's state models a commitment, then the creditor's state must also model the commitment—is of no relevance. An agent may adopt commitments towards other agents; however, if other agents do not expect it, those commitments are just harmless.

Our proposed definition of interoperability gives primacy to observations of each agent, i.e., the messages each sends and receives. We model communication between agents as being asynchronous and make only fundamental assumptions about it.

4.1.2 Commitment-Based Interoperability

We base our study of interoperability on Kant's distinction between constitutive and regulative rules, as developed by Searle [1995]. In simple terms, a constitutive rule specifies what action counts as what. For example, raising your hand may count as bidding in an outcry auction, or offering to give an answer if you are a student in a class. In this case, bidding or offering to answer are *institutional* actions. Similarly, a judge's specific

actions in the right context may count as creating a married couple. By contrast, a regulative rule constrains the performance of an action, e.g., that you cannot bid in an auction after a winner has been declared. In our approach, commitments are the key institutional facts, and the loci of institutional actions. Messages perform such actions by creating and manipulating commitments.

In our framework, message meanings are expressed as constitutive rules. The meaning of a message is specified in terms of its effects. The meaning may directly refer to commitments or indirectly affect commitments, as when the message counts as bringing about a condition of the commitment. For example, a price quote may constitute an offer to sell, treated as a (conditional) commitment. Each agent is described via its *constitutive specification*, which serves as an interface describing its assumptions. In essence, the constitutive specification of an agent tells us the meanings of the messages that the agent (presumably) respects. The intuition behind constitutive interoperability is that, in order to interoperate, the agents ought to agree about the institutional reality in which they exist. In other words, the agents agree on what their communications count as.

Constitutive interoperability is determined from constitutive specifications, that is, specifications that consist only of constitutive rules. If the interacting agents happen to apply mutually inconsistent constitutive rules, they would fail to interoperate. The above example where quote means different things to the customer and the merchant shows a violation of constitutive interoperability.

Message occurrence (when a particular message must be sent), ordering between the sends and receives of messages, and data flow among the messages are all regulative rules. A regulative specification may be viewed as encoding an agent's policies. For example, the merchant may have a regulatory rule that the customer must pay first in order for shipment to proceed. Regulative interoperability is determined from regulative specifications, that is, specifications that consist of regulative rules.

This paper concentrates on constitutive interoperability. In earlier work, we used $C+$ [Giunchiglia et al., 2004], an action description language, to specify and reason about protocols [Chopra and Singh, 2006a]. Here we employ a simpler language that is adequate for expressing constitutive rules and for reasoning about interesting cases of constitutive interoperability.

4.1.3 Contributions and organization

Our contributions in this paper are:

- A high-level definition of constitutive interoperability that takes into account the business meaning of communication, and that supports asynchronous communication.
- A language for constitutive specifications and a decision procedure for determining the constitutive interoperability of pairs of agents. A benefit of this approach is that it operates by program analysis rather than by building potentially large transition systems.

Section 4.2 presents our technical framework. Section 4.3 formalizes constitutive interoperability and provides a decision procedure for the same. The correctness proof is also provided. Section 4.4 places this work in the context of the literature.

4.2 Technical Framework

The framework consists of a language for constitutive specifications, an operational model of asynchronously communicating agents, and the formal semantics of the language in terms of the model.

4.2.1 Constitutive Specifications

Below, m_i range over messages; x, y, ... range over agents; p, q, ... range over propositions or Boolean formulas over them; $\top$ is the constant for truth; α is a propositional literal or its negation: identify α with $\neg\neg\alpha$. A commitment is a propositional letter. A commitment $\mathsf{C}(x, y, p, q)$ means that x is committed to y to bring about *consequent* q if *antecedent* p comes about.

Let's define our formal language via the following Backus-Naur Form productions. L is the starting symbol of our formal language. Below, Φ is a set of atomic propositions, $\mathcal{X}$ is a set of agent names, and *Message* names a message. We simplify the syntax by eliding parameters to concentrate on the points of interest here.

- $L \longrightarrow \{$*Message* means *Clause* $\}$

- *Clause* $\longrightarrow$ *Conjunction* | *Commitment*
- *Commitment* $\longrightarrow \mathsf{C}(\mathcal{X}, \mathcal{X}, \textit{Conjunction}, \textit{Disjunction})$
- *Conjunction* $\longrightarrow \Phi \mid \Phi \wedge \textit{Conjunction}$
- *Disjunction* $\longrightarrow \Phi \mid \Phi \vee \textit{Disjunction}$

As described in the above grammar, we restrict the antecedent and consequent of a commitment to be a conjunction and disjunction of propositional literals, respectively. This simplifies the presentation of the decision procedure for interoperability without a loss of expressiveness. For example (omitting agent names in commitments), m means $\mathsf{C}(p \vee q, r \wedge s)$ may be expressed as the four rules m means $\mathsf{C}(p, r)$, m means $\mathsf{C}(p, s)$, m means $\mathsf{C}(q, r)$, and m means $\mathsf{C}(q, s)$. Our grammar places an additional restriction that commitments may not be nested.

From a technical standpoint, an agent x's constitutive specification, $\mathbb{C}_x$, is a finite set of rules, each of the form of Schema 1.

Schema 1 m means p

The idea behind Schema 1 is to capture the *counts as* relationships that describe the institutional meanings of messages. In
Schema 1, the head p is a conjunction of propositional letters, and the body m is an action corresponding to a single message. When p is a commitment, the constitutive rule describes the creation of a commitment. Table 4.1 shows the constitutive specifications of a customer and merchant.

Table 4.1: Constitutive specifications of a customer and merchant

customer (c)
$\textit{Offer}(m, c)$ means $\mathsf{C}(m, c, \textit{pay}, \textit{goods})$
$\textit{Pay}(c, m)$ means $\textit{pay}$
$\textit{Goods}(m, c)$ means $\textit{goods}$
merchant (m)
$\textit{Offer}(m, c)$ means $\mathsf{C}(m, c, \textit{pay}, \textit{goods})$
$\textit{Pay}(c, m)$ means $\textit{pay}$
$\textit{Goods}(m, c)$ means $\textit{goods}$

We require that communicating agents have standard names and that their vocabularies are aligned. Thus we can talk coherently of the commitments in which each agent features. Specifically, if x and y are agents, and x refers to $\mathsf{C}(y, x, p, q)$, then we can compare this to $\mathsf{C}(y, x, p, q)$ as referred to by y. This assumption is not fundamental but simplifies our exposition.

4.2.2 Operational Semantics

Let x be an agent and $\mathbb{C}_x$ its constitutive specification. The formula $\langle m_0, \ldots, m_n \rangle \Vdash_x p$ means that the state of x after having observed $\langle m_0, \ldots, m_n \rangle$ models the proposition p. (Below, $p \vdash q$ means that we can derive q from p. When p and q are propositions, $\vdash$ is Boolean consequence.) Thus, $\Vdash_x$ is closed under the following rules of inference.

UNIT says that a message (by itself) always brings about the head of its defining constitutive rule.

$$\frac{m \text{ means } p \in \mathbb{C}_x}{\langle m \rangle \Vdash_x p} \qquad \text{(UNIT)}$$

PROP states that a proposition (that does not derive any commitments) holds if brought about by a message.

$$\frac{\langle m_n \rangle \Vdash_x p \qquad p \nvdash \mathsf{C}(r, s)}{\langle m_0, \ldots, m_n \rangle \Vdash_x p} \qquad \text{(PROP)}$$

HOLD states that a message that means a commitment brings it about unless the consequent of the commitment holds simultaneously. The consequent would cause the commitment to discharge. Thus a commitment may result only if it has not already and is not concurrently discharged (see below). A special case of this rule is when the antecedent is $\top$.

$$\frac{\langle m_0, \ldots, m_n \rangle \Vdash_x \neg q \qquad \langle m_n \rangle \Vdash_x \mathsf{C}(p, q)}{\langle m_0, \ldots, m_n \rangle \Vdash_x \mathsf{C}(p, q)} \qquad \text{(HOLD)}$$

DETACH explains the consequences of a commitment and its antecedent holding simultaneously. A stronger commitment, namely, with an antecedent of $\top$ comes to hold.

$$\frac{\langle m_0, \ldots, m_n \rangle \Vdash_x p \wedge \neg q \wedge \mathsf{C}(p, q) \qquad p \not\equiv \top}{\langle m_0, \ldots, m_n \rangle \Vdash_x \mathsf{C}(\top, q)} \qquad \text{(DETACH)}$$

SAT explains the satisfaction or discharge of a commitment. When the consequent of a commitment holds, the commitment is discharged and is thus active no more.

$$\frac{\langle m_n\rangle \Vdash_x q}{\langle m_0,\ldots,m_n\rangle \Vdash_x \neg \mathsf{C}(p,q)} \quad \text{(SAT)}$$

WEAKEN states that $\Vdash_x$ is closed under propositional derivation given by $\vdash$, as mentioned above.

$$\frac{\langle m_0,\ldots,m_n\rangle \Vdash_x p \qquad p \vdash q}{\langle m_0,\ldots,m_n\rangle \Vdash_x q} \quad \text{(WEAKEN)}$$

NEG states that $\Vdash_x$ deals with binary logic.

$$\frac{\langle m_0,\ldots,m_n\rangle \nVdash_x p}{\langle m_0,\ldots,m_n\rangle \Vdash_x \neg p} \quad \text{(NEG)}$$

INERTIA says that if an atomic proposition α holds and is not overturned by the next messaging action, then α continues to hold.

$$\frac{\langle m_0,\ldots,m_{n-1}\rangle \Vdash_x \alpha \qquad \langle m_n\rangle \nVdash_x \neg\alpha}{\langle m_0,\ldots,m_n\rangle \Vdash_x \alpha} \quad \text{(INERTIA)}$$

CMT describes the consequence relation between commitments. Of two commitments, the stronger commitment is the one whose antecedent is weaker or consequent is stronger.

$$\frac{\mathsf{C}(x,y,p_0,q_0) \qquad p_1 \vdash p_0 \qquad q_0 \vdash q_1}{\mathsf{C}(x,y,p_1,q_1)} \quad \text{(CMT)}$$

Thus CMT captures our intuition about covering the assumptions of agents. Let c_0 and c_1 be commitments. We say $c_0 \vdash c_1$—read as c_0 *covers* the assumptions of c_1—if and only if they have the same debtor and creditor, c_1's antecedent is stronger than c_0's, and c_1's consequent is weaker than c_0's. For an example where the antecedent is stronger, consider a customer c's commitment $c_0 = \mathsf{C}(c, m, \mathit{goods}, \mathit{pay})$ to a merchant m that c will pay if m sends the goods. Let's say the merchant assumes the commitment $c_1 = \mathsf{C}(c, m, \mathit{goods} \wedge \mathit{receipt}, \mathit{pay})$ from the customer instead. $c_0 \vdash c_1$ (c_0 covers c_1) because the customer's antecedent is weaker than the merchant's. For an example where the consequent is weaker, consider the customer's commitment $c_0 = \mathsf{C}(c, m, \mathit{goods}, \mathit{pay})$. Let the merchant's assumption be $c_1 = \mathsf{C}(c, m, \mathit{goods}, \mathit{pay} \vee \mathit{return})$. Clearly, $c_0 \vdash c_1$ because now the merchant's consequent is weaker than the customer's.

Lemma 1 $\Vdash_x$ terminates. ∎

Proof. (*Sketch*) Each inference rule for $\Vdash_x$ either reduces the sequence length or the depth of the formula being considered.

4.3 Constitutive Interoperability

First we present the definition of constitutive interoperability and then a decision procedure for determining it from agents' specifications. We present numerous examples along the way to illuminate the concepts involved.

4.3.1 Definition

Interoperability considers the prospect of interoperation. For this reason, it considers all possible enactments in which the agents may participate. Constitutive interoperability means that the agents would agree about whatever commitments as might result from any messages they might exchange. Thus it considers all potential observations of all agents, and fails if even one set of potential observations would cause failure of interoperation. Definition 6 considers only observations of creditors and debtors from the same quiescent vectors.

Definition 6 $\mathcal{A}$ is C-interoperable (written $[\![\langle\mathcal{A}\rangle]\!]$) iff

$$\forall \mathbb{O}_Q \in \mathcal{O}_{\mathcal{A}} : (\forall x, y : [O_x, O_y] = \mathbb{O}_Q :$$

$$(O_y \Vdash_y \mathsf{C}(x, y, p, q)) \rightarrow (O_x \Vdash_x \mathsf{C}(x, y, p, q)))$$

The idea is that if y's observations model, under its constitutive specification, that y is the creditor of commitment c, then any observations made by the debtor x must model, under its constitutive specification, that x is the debtor of c.

The above definition considers only quiescent vectors. The motivation for doing so is to provide an opportunity for the agents to "sync" up. In a distributed systems, the agents would in general observe different messages. Requiring that they agree upon their commitments without observing the same messages would in general lead to such a strong definition that would fail even when our intuition would be that the agents interoperate.

For example, say a merchant sends an offer to a customer that states: if you send me the payment, I will send you the goods. Say the customer receives this offer and sends the payment to the merchant. At this point, the customer has no knowledge of when the payment will arrive at the merchant; the merchant has no knowledge that the payment has been sent. The customer's observations legitimize the unconditional commitment on

part of the merchant to send it the goods. The merchant's observations do not. The above definition is not affected by this apparent discrepancy because it is only a transient discrepancy. When the payment arrives at the merchant, the commitment as expected by the customer will be covered by the merchant.

4.3.2 Decision Procedure

We introduce unique labels for the rules in a constitutive specification for easy reference in the text. For example, in $A =$
m means p, the label of the rule m means p is A, and we say that A *is* the rule m means p). Even though two rules in different constitutive specifications may have the same label, we use different labels throughout to avoid confusion.

Let A be a constitutive rule. $\hat{A}$ and $\breve{A}$ denote the head and body of A, respectively. Given a set of rules $\mathbb{A}$, $\widehat{\mathbb{A}} = \bigwedge_{A \in \mathbb{A}} \hat{A}$. To simplify the presentation, we introduce the empty rule ϵ as a member of every constitutive specification and its head as $\top$.

The intuition behind our decision procedure is to verify that each rule in the constitutive specification of an agent that would cause an agent to have a credit (a commitment in which the agent is the creditor) should be covered by a rule in the debtor's constitutive specification.

Example 14 In Table 4.2, rule Cus_1 which encodes the customer's assumption about an offer is supported by Mer_1 in the merchant's specification, that is, $Mer_1 \vdash Cus_1$. In fact, the merchant's commitment to send a receipt is not an assumption of the customer. ▮

Table 4.2: Offer

customer (c)
$Cus_1 = \mathit{Offer}$ means $\mathsf{C}(m, c, \mathit{pay}, \mathit{goods})$
merchant (m)
$Mer_1 = \mathit{Offer}$ means $\mathsf{C}(m, c, \mathit{pay}, \mathit{goods} \wedge \mathit{receipt})$

Example 15 With reference to Table 4.2, if the merchant's specification had $Mer_2 = \mathit{Offer}$ means $\mathsf{C}(m, c, \mathit{pay}, \mathit{receipt})$ instead of Mer_1, then $Mer_2 \nvdash Cus_1$ and when the *Offer* message is exchanged, it will cause a commitment misalignment. ▮

There is an additional caveat though: the agents must also agree on the messages that affect a commitment. This means that the decision procedure must check the agents' specifications for the respective compatibility of the rules that bring about the antecedent and consequent of a commitment. Specifically, the debtor should cover the ways in which the creditor may bring about the antecedent, and the creditor should cover all the ways in which the debtor may bring about the consequent. In Table 4.2 there are no such rules, hence the agents vacuously agree.

Example 16 Let's consider the agents in Table 4.3. Clearly, $Mer_3 \vdash Cus_2$. In addition, the merchant covers all the ways in which a customer expects to make a payment (Cus_3 is covered by Mer_4) therefore ensuring that when the customer pays, the merchant understands that. Similarly, the customer understands all the ways the merchant can cause the *goods* condition to hold (Mer_6 is covered by Cus_4). ▮

Table 4.3: Offer with antecedent and consequent rules

customer (c)
$Cus_2 = Offer$ means $\mathsf{C}(m, c, pay, goods)$
$Cus_3 = PayCash$ means pay
$Cus_4 = GoodsShip$ means $goods$
$Cus_5 = GoodsExpedited$ means $goods$
merchant (m)
$Mer_3 = Offer$ means $\mathsf{C}(m, c, pay, goods)$
$Mer_4 = PayCash$ means pay
$Mer_5 = PayCredit$ means pay
$Mer_6 = GoodsShip$ means $goods$

Example 17 Referring to Table 4.3, suppose that the merchant did not have the rule Mer_4 meaning she only accepts credit cards. Then upon doing $PayCash$, the customer's state would model the commitment $\mathsf{C}(m, c, \top, goods)$ (because of DETACH); however the merchant's state would not model $\mathsf{C}(m, c, \top, goods)$ upon receiving $PayCash$. Hence, interoperability fails. ▮

Definitions 7 and 8 introduce the machinery necessary to formalize this caveat. Definition 7 introduces the notion of an *antecedent predecessor*. Let $A = m\mathsf{means}\mathsf{C}(x, y, p, q) \in$

$\mathbb{C}$. An antecedent predecessor of A is a subset of $\mathbb{C}$ such that the rules in the predecessor explain the causation of each propositional letter in p. The set of all antecedent predecessors of an rule A is denoted by $\mathbb{IP}_A$.

Example 18 In Table 4.3, the commitment in Cus_2 has only one propositional letter *pay*. An antecedent predecessor of Cus_2 is $\{Cus_3\}$. Mer_3 has two antecedent predecessors: $\{Mer_4\}$ and $\{Mer_5\}$. Thus, $\mathbb{IP}_{Cus_2} = \{\{Cus_3\}\}$ and $\mathbb{IP}_{Mer_3} = \{\{Mer_4\}, \{Mer_5\}\}$. ∎

Example 19 In Table 4.2, $\mathbb{IP}_{Cus_1} = \{\}$, $\mathbb{IP}_{Mer_1} = \{\}$. ∎

No subset of an antecedent predecessor of a rule A should itself be an antecedent predecessor of A because that means the former contains rules not relevant to the antecedent. What is not relevant to a commitment will necessarily have no effect on commitment-level interoperability.

Definition 7 Let $A = m\mathsf{meansC}(x, y, p, q)$ be a rule in $\mathbb{C}$. Then the *antecedent predecessors* of A in $\mathbb{C}$ denoted by $\mathbb{IP}_A$ is defined as

$$\{\Delta | \Delta \subseteq \mathbb{C} : (\widehat{\Delta} \equiv p \textit{ and } \neg(\exists \Delta' : \widehat{\Delta}' \equiv p \textit{ and } \Delta' \subset \Delta))\}$$

Recall that the consequent of a commitment is a disjunction of propositional literals. Bringing any one of those about satisfies the commitment. Unlike the antecedent predecessor which is a set of rules, a consequent predecessor is a single rule whose head derives at least one propositional literal in the consequent.

Definition 8 Let $A = m$ means $\mathsf{C}(x, y, p, q)$ be a rule in $\mathbb{C}$ where q is the disjunction $q_0 \vee \ldots \vee q_n$ of propositional letters. The *consequent predecessors* of A in $\mathbb{C}$ denoted by $\mathbb{C}_A$ is defined as

$$\{R | R \in \mathbb{C} \text{ and } \exists q_i \ (0 \leq i \leq n) \ : \hat{R} \vdash q_i\}$$

Example 20 In Table 4.2, $\mathbb{C}_{Cus_1} = \{\}$, $\mathbb{C}_{Mer_1} = \{\}$. ∎

Example 21 Referring to Table 4.3, $\mathbb{C}_{Cus_2} = \{Cus_4, Cus_5\}$, $\mathbb{C}_{Mer_3} = \{Mer_6\}$. ∎

Definition 9 finally puts together all the elements discussed above in defining the *complete coverage* of a rule that causes a commitment credit. For such a rule to be completely covered, the following conditions must be satisfied:

1. *Rule coverage*: The debtor must cover the rule: a credit represents an assumption of the creditor.

2. *Antecedent coverage*: The debtor must cover all the ways in which the creditor may bring about the antecedent of the commitment—these represent the assumptions of the creditor. Additionally, it means that if the creditor expects to deal in n distinct messages to bring about the antecedent of a commitment, then the debtor's cover cannot involve more than those n messages.

3. *Consequent coverage*: The creditor must cover all the ways in which the debtor may bring about the consequent of the commitment—these represent the assumptions of the debtor. This ensures that any message that can discharge a commitment on the debtor's side will also discharge the commitment on the creditor's side.

Definition 9 Let $\mathbb{C}_x$ and $\mathbb{C}_y$ be the constitutive specifications of agents x and y, respectively. Let $E \in \mathbb{C}_y$ be
m means $\mathsf{C}(x, y, p, q)$. E is *completely covered*, denoted by $\lfloor E \rfloor$ iff E is covered, that is, $\exists M \in \mathbb{C}_x : (M \vdash E)$ and the following hold:

- Antecedent coverage: $\forall \mathbb{S} \in \mathbb{IP}_E : \exists \mathbb{V} \in \mathbb{IP}_M : (\bigcup_{A \in \mathbb{V}} \breve{A}) \subseteq (\bigcup_{A \in \mathbb{S}} \breve{A})$
- Consequent coverage: $\forall S \in \mathbb{C}_M : \exists V \in \mathbb{C}_E : \breve{S} \equiv \breve{V}$

In Definition 9 above, each $\mathbb{S}$ referred to in the *antecedent coverage* clause represents a "way" (assumption) of the creditor that must be covered by the debtor. Similarly, each S in the *consequent coverage* clause represents a "way" of the debtor that must be covered by the creditor.

Definition 10 Let $\mathcal{A}$ be a two agent system. $\mathcal{A}$ is *compatible*, denoted by $[[\mathcal{A}]]$, iff

$$\forall y \in \mathcal{A} : \forall E = m \text{ means } \mathsf{C}(x, y, p, q) \in \mathbb{C}_y : \lfloor E \rfloor$$

Algorithm 1 is pseudo code for Definition 10.

Figure 4.1 and 4.2 show the program analysis graphs for the agents in Table 4.2 and 4.3, respectively. A program analysis graph is constructed as follows. For each agent, for each rule E in which a credit is created, create a circle labeled with E. Denote each of

Algorithm 1: Algorithm for determining $[[\mathcal{A}]]$

```
1  foreach (agent y in a two agent system) do
2      foreach (E in y's specification which means a credit C(x, y, p, q) for y)
       do
3          if (there exists an M in x's specification such that M covers E) then
4              foreach (way in which y assumes p can hold) do
5                  if (x covers that way) then
6                      continue;
7                  else
8                      return false;
9              foreach (way in which x assumes q can hold ) do
10                 if (y covers that way) then
11                     continue;
12                 else
13                     return false;
14         else
15             return false;
16 return true;
```

its antecedent predecessors by a dotted box connected to the circle by an arrow labeled P. Label the box with the rules in that antecedent predecessor. Denote each of its consequent predecessors by a dotted box connected to the circle by an arrow labeled with C. If a rule M in the other agent covers E ($M \vdash E$), then create a circle labeled M and connect M to E with an arrow directed towards E. Indicate M's antecedent and consequent predecessors as described for E. If there exists an antecedent predecessor of M that covers one of E, draw a directed arrow from the former to the latter. Similarly, if there exists a consequent predecessor of E that covers one of M, draw a directed arrow from the former to the latter. If at the end of this graph construction, E is not connected to some M, or if one of E's antecedent predecessors is not connected to some of M's, or if one of M's consequent predecessors is not connected to some of $E's$, then the agents are not compatible.

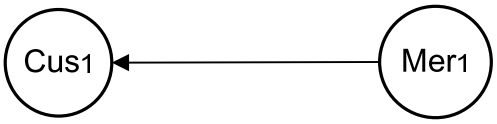

Figure 4.1: Program analysis graph for agents in Table 4.2

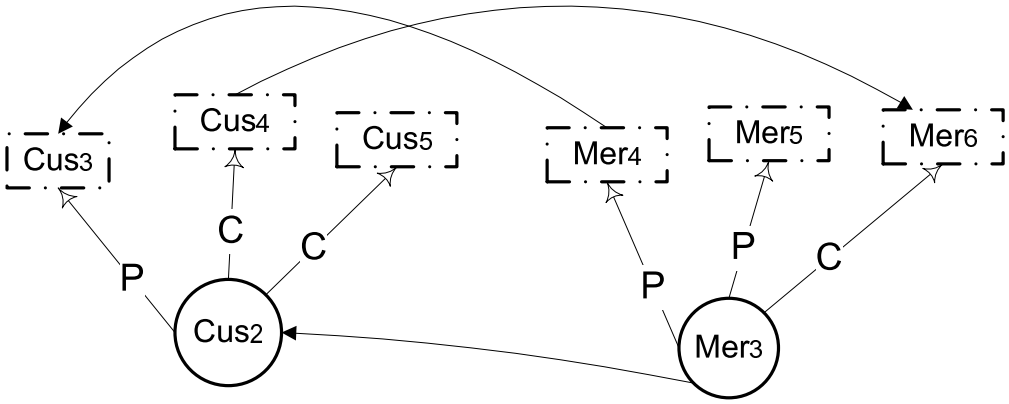

Figure 4.2: Program analysis graph for agents in Table 4.3

Example 22 Consider the agents in Table 4.4. Here the merchant ships goods to customers of legal age only. However, she accepts payment by credit card to be proof of legal age. The customer, however, provides her birth date as proof of legal age. First, $Mer_7 \vdash Cus_6$.

$\mathbb{IP}_{Cus_6} = \{\{Cus_7, Cus_8\}\}$, $\mathbb{IP}_{Mer_7} = \{\{Mer_8\}\}$, and the set of messages involved in $\{Mer_8\}$ is a subset of the messages involved in $\{Cus_7, Cus_8\}$ ($\{PayCredit\} \subseteq \{PayCredit, ProvideBirthDate\}$). Therefore, $\lfloor Cus_6 \rfloor$. ∎

Table 4.4: Antecedent coverage: merchant uses fewer messages

customer (c)
$Cus_6 = Offer$ means $\mathsf{C}(m, c, pay \wedge legalAge, goods)$
$Cus_7 = PayCredit$ means pay
$Cus_8 = ProvideBirthDate$ means $legalAge$
merchant (m)
$Mer_7 = Offer$ means $\mathsf{C}(m, c, pay \wedge legalAge, goods)$
$Mer_8 = PayCredit$ means $pay \wedge legalAge$

Figure 4.3 shows the program analysis graph for the agents in Table 4.4.

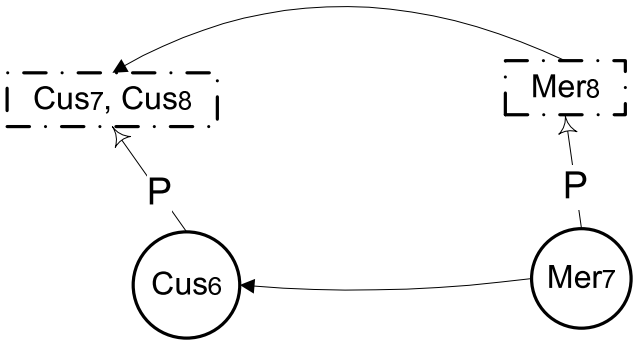

Figure 4.3: Program analysis graph for agents in Table 4.4

Example 23 Referring to Table 4.5, $\mathbb{IP}_{Cus_9} = \{\{Cus_{10}\}\}$ and $\mathbb{IP}_{Mer_9} = \{\{Mer_{10}, Mer_{11}\}\}$. The messages involved in
$\{Mer_{10}, Mer_{11}\}$ are not a subset of $\{Cus_{10}\}$. Therefore, when customer does $PayCredit$ (after $Offer$), she assumes the commitment $\mathsf{C}(m, c, \top, goods)$ whereas the merchant does not because it does not see $PayCredit$ to mean $legalAge$: it also expects to observe $ProvideBirthDate$. Therefore, antecedent coverage for $\{Cus_{10}\}$ does not hold. Therefore, $\lfloor Cus_9 \rfloor$ does not hold. ∎

Table 4.5: No antecedent coverage: customer uses fewer messages

customer (c)
$Cus_9 = Offer$ means $\mathsf{C}(m, c, pay \wedge legalAge, goods)$
$Cus_{10} = PayCredit$ means $pay \wedge legalAge$
merchant (m)
$Mer_9 = Offer$ means $\mathsf{C}(m, c, pay \wedge legalAge, goods)$
$Mer_{10} = PayCredit$ means pay
$Mer_{11} = ProvideBirthDate$ means $legalAge$

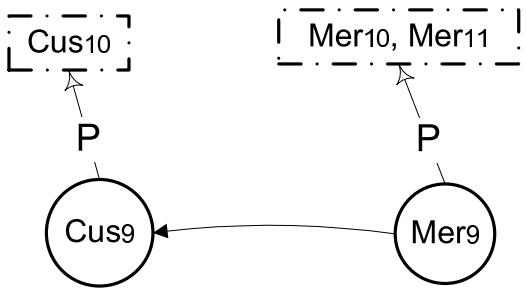

Figure 4.4: Program analysis graph for agents in Table 4.5

Figure 4.4 shows the program analysis graph for the agents in Table 4.5.

Example 24 Referring to Table 4.6, $\lfloor Cus_{11} \rfloor$ holds in spite of the fact that between the two agents the meanings of $PayCredit$ and $ProvideBirthDate$ are interchanged. ∎

Example 25 A merchant may unconditionally commit to sending the goods. Referring to Table 4.7, $Mer_{15} \vdash Cus_{14}$, $\mathbb{IP}_{Cus_{14}} = \{\{Cus_{15}\}\}$, $\mathbb{IP}_{Mer_{15}} = \{\epsilon\}$ and since causing ϵ requires no messages, we obtain $\lfloor Cus_{14} \rfloor$. ∎

Example 26 For the sake of completeness, let's consider an example that bring consequent coverage into focus. Referring to Table 4.8, $Mer_{16} \vdash Cus_{16}$, and $\mathbb{C}_{Cus_{16}} = \{Cus_{17}, Cus_{18}\}$ and $\mathbb{C}_{Mer_{16}} = \{Mer_{17}\}$. When the merchant sends $GoodsShip$ and discharges her commitment, the customer also understands its receipt to mean discharge of the merchant's commitment. Thus, $\lfloor Cus_{16} \rfloor$ holds. ∎

Table 4.6: Offer with jumbled but adequate meanings

customer (c)
$Cus_{11} = \mathit{Offer}$ means $\mathsf{C}(m, c, \mathit{pay} \wedge \mathit{legalAge}, \mathit{goods})$
$Cus_{12} = \mathit{PayCredit}$ means $\mathit{legalAge}$
$Cus_{13} = \mathit{ProvideBirthDate}$ means pay
merchant (m)
$Mer_{12} = \mathit{Offer}$ means $\mathsf{C}(m, c, \mathit{pay} \wedge \mathit{legalAge}, \mathit{goods})$
$Mer_{13} = \mathit{PayCredit}$ means pay
$Mer_{14} = \mathit{ProvideBirthDate}$ means $\mathit{legalAge}$

Table 4.7: Making an unconditional commitment

customer (c)
$Cus_{14} = \mathit{Offer}$ means $\mathsf{C}(m, c, \mathit{pay}, \mathit{goods})$
$Cus_{15} = \mathit{PayCash}$ means pay
merchant (m)
$Mer_{15} = \mathit{Offer}$ means $\mathsf{C}(m, c, \top, \mathit{goods})$

Figure 4.5 shows the program analysis graph for the agents in Table 4.8.

Example 27 Also, it is worth considering the agents in Table 4.9. Even though *GoodsShip* means different things to the customer and merchant, when they observe the message, it discharges the merchant's commitment in both the customer and merchant's model. Here too, we have $\lfloor Cus_{19} \rfloor$. ∎

Figure 4.6 shows the program analysis graph for the agents in Table 4.9.

Table 4.8: Consequent coverage

customer (c)
$Cus_{16} = \mathit{Offer}$ means $\mathsf{C}(m, c, \mathit{pay}, \mathit{goods} \vee \mathit{refund})$
$Cus_{17} = \mathit{GoodsShip}$ means goods
$Cus_{18} = \mathit{RefundMoney}$ means refund
merchant (m)
$Mer_{16} = \mathit{Offer}$ means $\mathsf{C}(m, c, \mathit{pay}, \mathit{goods})$
$Mer_{17} = \mathit{GoodsShip}$ means goods

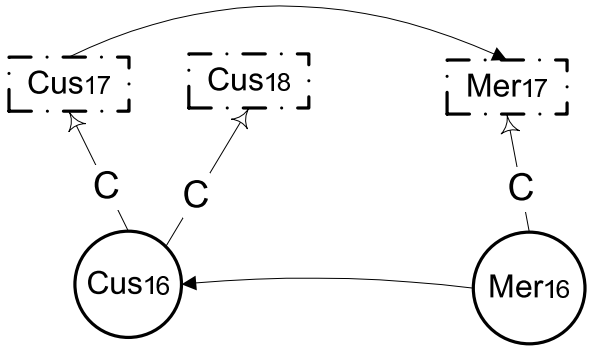

Figure 4.5: Program analysis graph for agents in Table 4.8

Table 4.9: Consequent coverage: case of adequate meaning

customer (c)
$Cus_{19} = Offer$ means $\mathsf{C}(m, c, pay, goods \vee refund)$
$Cus_{20} = GoodsShip$ means $refund$
merchant (m)
$Mer_{18} = Offer$ means $\mathsf{C}(m, c, pay, goods)$
$Mer_{19} = GoodsShip$ means $goods$

Theorem 2 $[\![\{x, y\}]\!]$ if and only if $[[\{x, y\}]]$. ∎

Proof. (*Sketch*) The proof is by induction on the length of quiescent vectors. For quiescent observation vectors of length 1, any commitment that exists must be caused by rules (in the creditor's theory) pertaining to a single message. As a result, our decision procedure would find the rules (in the debtor's theory) that cover such creditor rules. Conversely, for any pair of covering rules the message mentioned in their bodies could be observed to be sent and received. Thus if the decision procedure finds a cover, there would be an observation vector where that is realized.

Now assume that the theorem holds for quiescent observation vectors of length up to k. Consider any quiescent observation vector of length $k + 1$. Any commitment that holds after a sequence of length $k + 1$ either held at the end of the first k observations in that sequence, or is caused by the $(k + 1)^{st}$ observation. In the former case, the inductive hypothesis applies. In the latter, the rules for the last message apply. The antecedent and consequent supports for these rules must have already have been accounted for in the first

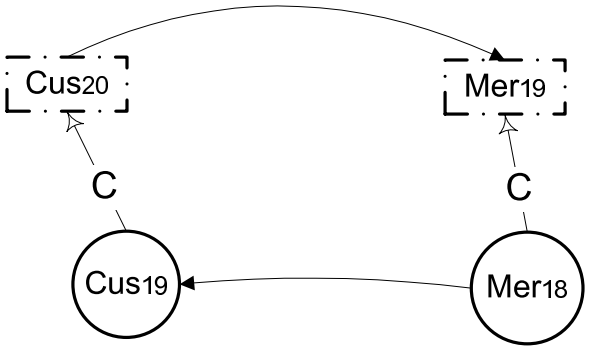

Figure 4.6: Program analysis graph for agents in Table 4.9

k observations. Hence, by induction, the result holds.

4.4 Discussion

Approaches based on verifying compliance at runtime [Alberti et al., 2004], [Venkatraman and Singh, 1999] are important in the context of open systems since agents may behave in unpredictable ways; also it is necessary to have independent arbiters in cases of dispute. Alberti *et al.* [2007] present SCIFF, an abductive reasoning framework for reasoning about the policies of services with the purpose of verifying if a goal might be reached. In that sense, SCIFF is similar to specifying agents in $C+$ [Giunchiglia et al., 2004], and running queries in $CCalc$, which is the reasoning tool that implements $C+$.

Our work falls within the broader context of normative multiagent systems (for example, [Boella and van der Torre, 2004]). Our use of constitutive rules to means *counts as* is decidedly narrow in that a message only counts as meaning something for a particular agent, and not in the context of the institution the agent acts in. In this work, we assume that the agents act in an institution, but may have differing views on the creation of institutional facts. Such differences in views are the basis of failure of interoperability. In future work, we will broaden the formalization to include institutions.

Winikoff [2007] studies the distributed enactment of a commitment protocol amongst agents. Commitments are mapped to BDI plans, and all possible plans are checked to see if they allow making progress towards desirable goal states. This enables designers to specify commitment protocols and not have to worry about low-level messaging details, which is

highly desirable. Since commitments are already aligned in a commitment protocol, there is no need to check commitment-level interoperability between the distributed commitment machines, which is the question we address in this work.

We will extend our algorithm to handle additional commitment operations such as delegate, cancel, assign, and release. Doing so will enable modeling general multiparty interactions. Addressing regulative interoperability in more depth is also an important direction.

Chapter 5

Discussion

In Chapter 1, we motivated commitments as an important abstraction for modeling business processes. Commitments support a semantic notion of compliance, which in turns enables flexible business process enactment by agents. We then motivated commitment alignment as a key form of interoperability, failing which interaction among agents would break down. We also discussed the causes of misalignment: asynchrony, distribution, and heterogeneity. In Chapter 2, we formally characterized commitment alignment in terms of the inferences agents make from their observation. In Chapter 3, we addressed autonomy and distribution. We did that by formalizing commitment operations in distributed settings. And finally, in Chapter 4, we addressed heterogeneity.

Now, we discuss our work in a broader context, and lay out some future directions.

5.1 Commitments and Agent Communication

Commitments have been studied in the context of distributed problem solving and coordination. Bratman [1992] points out the importance of a agents committing to activities for successfully carrying out a shared cooperative activity. Grosz and Kraus [1996] formulate shared plans for coordinating group action. In their framework, an agent can adopt two types on intentions, intend-to and intend-that, that commit the agent to an action and state of affairs respectively. Jennings [1993] presents commitments as a fundamental notion for efficient coordination in distributed systems. Jennings also mentions conventions which monitor the commitments and state when a commitment may be reassessed. Jennings

further reformulates different models of coordination in terms of commitments. Shoham's agent oriented computing paradigm [1993] introduces obligation as a modality required to describe the mental state of an agent. A common feature of all of the above work is that they present commitments as a mentalistic notion, purely for the purposes of coordination.

Castelfranchi [1993] first presented commitments as a social relation, that is, as a relation between agents, with the aim of understanding organizational activity. That work led to a significant result in agent communication. In the mid nineties, a major challenge in agent communication was the formalization of communication among agents. Most approaches formalized communication in terms of mental agency, that is, in terms of beliefs and intentions (for an example, see [Finin et al., 1994]). Singh [1998] pointed out a fundamental shortcoming in such formalizations: since the mental states of agents are private, and therefore not verifiable, it is not possible to ascertain what agents mean by their communications. Instead, Singh proposed that communication should be given a social semantics, one based on commitments. For a survey of the state of the art in agent communication, see [Dignum et al., 2006].

Singh introduced the commitment operations in [1999]. Formalization of commitment operations based on Singh's presentation are found in [Yolum and Singh, 2002; Chopra and Singh, 2004; Desai et al., 2007b]. Fornara and Colombetti [2002] propose an object-oriented formalization of commitments. Singh [2008] delineated the model-theoretic semantics of commitments.

5.2 Software Engineering

First and foremost, commitment capture the business meaning of interactions as opposed to low-level operational representations of interaction. In this sense, commitments are a useful software engineering abstraction. Yolum and Singh [2002] first formalized commitment protocols, and showed how such protocols enabled flexible execution. Desai *et al.* [2005] showed how to construct agents from representations of commitment protocols. Mallya and Singh [2007] showed how to reason about subsumption among commitment protocols. Such reasoning is essential to understanding that payment by credit card is a special kind of payment protocol. Composition of commitment protocols is studied in [Desai and Singh, 2007].

The question of the *enactability* of a protocol is an important one [Fu et al., 2004; Desai and Singh, 2008; Carbone et al., 2006]. A protocol is enactable if and only if distribution of the protocol among agents supports exactly those executions that are supported when the protocol is considered as a whole. For example, the protocol specification may state that a message m from agent x to y must happen immediately after m' from z to y. Clearly, this constraint cannot be enforced in a distributed system: x cannot see m'. Hence, we would say that such a protocol is not enactable. Since interoperability is at the level of agents and their interfaces, this dissertation does not consider problems of enactability.

5.2.1 Architecture and Patterns

An architectural style specifies a family of configurations of *components* and *connectors* subject to stated *constraints* [Shaw and Garlan, 1996].

In these terms, existing SOAs are an architectural style whose major components are (service) provider and consumer, and whose connector is a protocol by which a consumer invokes a provided service. (For simplicity, we ignore registries, and service publication and discovery.) A practical SOA includes specialized components and connectors, such as for resource management and other enterprise functions (identity, billing, and so on), and imposes additional constraints so appropriate components interoperate with the management and enterprise components.

Recently, Singh *et al.* [2008] have proposed Commitment-Based SOA (CSOA) as an architectural style whose components are the participants in a service engagement, and whose connectors are patterns of commitment operations.

Table 5.1 contrasts commitment-based and existing SOAs. Thus, CSOA is not a unique style but has many flavors depending on the patterns selected. Such flexibility is necessary to support the nuances of service engagements. The primary constraint on a sound implementation of CSOA is that at runtime all commitments are eventually handled in one way or another.

Model-Driven Architecture (MDA) provides a useful way to think of the relationship between CSOA and existing SOAs. In MDA terms, CSOA is a Computation Independent Model whereas existing SOAs are Platform Independent Models. In other words, the move to CSOA would represent the step—often repeated in computer science—of moving from lower to higher abstractions. Because commitments are computation independent, yet

Table 5.1: A comparison of commitment-based and existing SOAs

Elements	**Existing SOAs**	**Commitment-Based SOA**
Components	Service provider and consumer	Business service provider and consumer agents
Connectors	Operations and message patterns (in, out, in-out, out-in)	Commitment patterns
Invariants	Match operation and message signatures	Each party knows only the identity of parties with whom it features in a commitment
Model	Control and data flow	Operations on commitments

lend themselves to rigorous operationalization, CSOA can help bridge the well-recognized gap between business and IT [Smith and Fingar, 2002].

Others have begun to recognize the importance of high-level abstractions, but their work still employs operational abstractions. We briefly describe some of these below.

Benatallah *et al.* propose patterns called *business-level interfaces and protocols* [2006]. However, their patterns ignore business meanings (like CDL and BPEL), thereby leading to rigid interoperation. For example, if a message interface specifies that a customer should make a payment subsequent to the receipt of goods, then a service realizing such an interface must behave accordingly. It ought not to take any liberties such as reversing the order of the messages, interposing other messages, or introducing another party such as a payment agency. But, real-life service engagements typically presume such flexibility. Limiting flexibility subverts the services vision because it creates avoidable friction in the web of value.

Kumaran [2004] presents four abstraction layers for enterprise modeling: strategy (business considerations), operation (business functions conceptualized via tasks and artifacts), execution (analogous to existing SOAs), and implementation. CSOA would help extend Kumaran's operation layer to multienterprise service engagements, and commitment patterns would provide richer representations that facilitate modeling enterprise operations perspicuously and reusably.

Bhattacharya *et al.* [2007] propose an artifact-based approach, where an artifact is a business document such an *Order*. Each artifact has its own lifecycle. Again, the problem here is that the lifecycles have no semantic basis, and therefore turn out to be unnecessarily rigid.

van der Aalst *et al.* [2003] document patterns commonly used in workflow modeling such as branching and synchronization; Hohpe and Woolf [2003] describe several enterprise integration styles and patterns; Barros *et al.* [2005] describe low-level messaging patterns used in services. No doubt, all these patterns encode valuable experience and are widely applicable; however, none of these represent business-level patterns.

Burgess [2005] proposes *promises*, which are in a pragmatic sense similar to commitments, as a useful abstraction for understanding distributed systems of autonomous agents.

5.2.2 Software Components and Interoperability

Our solution for addressing heterogeneity is inspired from research in the area of software components. The interoperability of independently designed components has been an issue of long standing interest in components research. Interfaces form the basis of interoperation in such approaches. Allen and Garlan [1997] formally introduced the idea that the connectors between components should be formalized as protocols that describe constraints on messaging. Yellin and Strom [1997] formalized interfaces as finite state machines, with messages occurring along transitions. Given two components with such interfaces, Yellin and Strom gave the conditions for compatibility between the interfaces. If the interfaces were found to be compatible, then the components were deemed interoperable. Yellin and Strom assume synchronous communication because of undecidability results under asynchronous communication [Brand and Zafiropulo, 1983]. Following Yellin and Strom, there have been numerous alternative formalizations of component interoperability (for example, see [de Alfaro and Henzinger, 2001; Canal et al., 2003; Chopra and Singh, 2006b; 2007]).

So far, researchers have approached interoperability from the point of view of coordination: their definitions are couched in terms of operational notions such as choice and deadlock-freedom of the components. Such formalizations are no doubt relevant and essential; however, they do not capture the business meaning of interaction. Our commitment-based approach addresses this shortcoming. It abstracts away from the operational notions

of interoperability, and makes commitment alignment the sole criterion. Our vision is that designers first specify agents in terms of commitments, check for commitment alignment, and then successively refine the specifications in a model-driven manner so as to obtain implementations that also meet the more operational notions of interoperability.

Related to the notion of interoperability is that of conformance. Broadly speaking, conformance is a property preserving substitution relation—the property of interest to us is interoperability. This saves the hassle of checking interoperability at a system-wide level every time a component is replaced. Conformance is discussed in [Endriss et al., 2003; Fournet et al., 2004; Baldoni et al., 2006; Bravetti and Zavattaro, 2007; Castagna et al., 2008]. In this dissertation, we presented commitment alignment as a key form of interoperability; it would be interesting to formalize conformance in terms of commitments.

5.3 Future Work

Here, we outline some of the more promising directions of future work.

5.3.1 Metacommitments

An important direction is to extend the message language presented in Chapter 2 to handle metacommitments, that is, commitments about commitments. Such commitments often arise in real situations. For example, Alice commits to Bookie that if Bookie commits to sending her the book, then Alice will pay, meaning $\mathsf{C}(\mathit{Alice}, \mathit{Bookie}, \mathit{created}(\mathit{Bookie}, \mathit{Alice}, \top, \mathit{book}), \mathit{pay})$. One of the conditions for a successful delegation could be the commitment $\mathsf{C}(\mathit{delegatee}, \mathit{delegator}, \mathit{delegated}(\mathit{delegator}, \mathit{creditor}, r, u, \mathit{delegatee}), \mathit{created}(\mathit{delegatee}, \mathit{creditor}, r, u))$, that is, the delegatee commits to the delegator that it will accept the delegation. Another common kind of metacommitment involves compensation for cancellations. Often this involves a third party. For example, Bookie may be operating in a marketplace Pamazon, which makes the commitment to Alice that if Bookie cancels its commitment for the book after payment has been made, then Bookie will commit to refund Alice, meaning that $\mathsf{C}(\mathit{Pamazon}, \mathit{Alice}, \mathit{cancelled}(\mathit{Bookie}, \mathit{Alice}, \top, \mathit{book}), \mathit{created}(\mathit{Bookie}, \mathit{Alice}, \top, \mathit{refund}))$.

5.3.2 Richer Interface Language

The language for agent interfaces presented in Chapter 4 is simple; it does not support CANCEL, RELEASE, DELEGATE, and ASSIGN. Now that we have solved alignment in distributed settings, an obvious and important next step is to support all the commitment operations in the interface specification language.

5.3.3 Tools and Middleware for Enterprises

Our solutions for autonomy and distribution could be packaged into a middleware that sits beneath agents. To make sure that agents would not get misaligned, the most designers would have to do is simply run a tool that checks for interface compatibility.

5.3.4 Pattern Language

When it comes to business process modeling, the focus in computing has shifted undeniably from orchestration to interactions. In fact, industry-supported orchestration languages such as BPEL have been noted to be completely unnecessary [van der Aalst et al., 2005]. Interaction has now begun to be treated as a first-class abstraction in high-level programming languages such as Java [Hu et al., 2008]. A language for encoding commitment patterns, such as those reported in [Singh et al., 2008], would be highly desirable. The semantics of the language itself would be rooted in the results of this dissertation.

Bibliography

Marco Alberti, Davide Daolio, Paolo Torroni, Marco Gavanelli, Evelina Lamma, and Paola Mello. Specification and verification of agent interaction protocols in a logic-based system. In *Proceedings of the 19th ACM Symposium on Applied Computing*, pages 72–78, 2004.

Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, Marco Montali, and Paolo Torroni. Web service contracting: Specification and reasoning with SCIFF. In *Proceedings of the 4th European Semantic Web Conference*, pages 68–83, 2007.

Robert Allen and David Garlan. A formal basis for architectural connection. *ACM Transactions on Software Engineering and Methodology*, 6(3):213–249, 1997.

Leila Amgoud, Nicolas Maudet, and Simon Parsons. An argumentation-based semantics for agent communication languages. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 38–42, 2002.

Matteo Baldoni, Cristina Baroglio, Alberto Martelli, and Viviana Patti. A priori conformance verification for guaranteeing interoperability in open environments. In *Proceedings of the 4th International Conference on Service-Oriented Computing*, pages 339–351, December 2006.

Alistair Barros, Marlon Dumas, and Arthur H.M. ter Hofstede. Service interaction patterns. In *Business Process Management*, volume 3649 of *LNCS*, pages 302–318. Springer, 2005.

Boualem Benatallah, Fabio Casati, and Farouk Toumani. Analysis and management of web service protocols. In *Conceptual Modeling ER 2004*, volume 3288 of *LNCS*, pages 524–541. Springer, 2004.

Boualem Benatallah, Fabio Casati, Farouk Toumani, Julien Ponge, and Hamid R. Motahari Nezhad. Service Mosaic: A model-driven framework for web services life-cycle management. *IEEE Internet Computing*, 10(4):55–63, 2006.

Jamal Bentahar, Bernard Moulin, John-Jules Ch. Meyer, and Brahim Chaib-draa. A logical model for commitment and argument network for agent communication. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems*, pages 792–799, 2004.

Kamal Bhattacharya, Cagdas Gerede, Richard Hull, Rong Liu, and Jianwen Su. Towards formal analysis of artifact-centric business process models. In *Business Process Management*, volume 4714 of *LNCS*, pages 288–304. Springer, 2007.

Guido Boella and Leendert W. N. van der Torre. Regulative and constitutive norms in normative multiagent systems. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR)*, pages 255–266. AAAI Press, 2004.

Rafael H. Bordini, Mehdi Dastani, and Michael Winikoff. Current issues in multi-agent systems development. In *Engineering Societies in the Agents World VII*, volume 4457 of *Lecture Notes in Computer Science*, pages 38–61. Springer, 2007.

BPEL. Business process execution language for web services, version 1.1, May 2003. www-106.ibm.com/developerworks/webservices/library/ws-bpel.

BPMN. Business process modeling notation, v1.1, January 2008. http://www.bpmn.org/.

Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.

Michael E. Bratman. Shared cooperative activity. *The Philosophical Review*, 101:327–341, 1992.

Mario Bravetti and Gianluigi Zavattaro. Towards a unifying theory for choreography conformance and contract compliance. In *6th International Symposium on Software Composition*, volume 4829 of *LNCS*, pages 34–50. Springer, 2007.

Mark Burgess. An approach to understanding policy based on autonomy and voluntary cooperation. In *Ambient Networks*, volume 3775 of *LNCS*, pages 97–108. Springer, 2005.

Carlos Canal, Lidia Fuentes, Ernesto Pimentel, Jos M. Troya, and Antonio Vallecillo. Adding roles to CORBA objects. *IEEE Transactions on Software Engineering*, 29(3): 242–260, 2003.

Marco Carbone, Kohei Honda, Nobuko Yoshida, Robin Milner, and Steve Ross-Talbot. A theoretical basis of communication-centered concurrent programming, October 2006. http://www.w3.org/2002/ws/chor/edcopies/theory/note.pdf.

Giuseppe Castagna, Nils Gesbert, and Luca Padovani. A theory of contracts for web services. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 261–272, 2008.

Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In *Proceedings of the AAAI-93 Workshop on AI and Theories of Groups and Organizations: Conceptual and Empirical Research*, 1993.

Cristiano Castelfranchi. Modelling social action for AI agents. *Artificial Intelligence*, 103 (1-2):157–182, 1998.

Amit Chopra and Munindar P. Singh. Nonmonotonic commitment machines. In Frank Dignum, editor, *Advances in Agent Communication: Proceedings of the 2003 AAMAS Workshop on Agent Communication Languages*, volume 2922 of *LNAI*, pages 183–200. Springer-Verlag, 2004.

Amit K. Chopra and Munindar P. Singh. Constitutive interoperability. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pages 794–804, 2008.

Amit K. Chopra and Munindar P. Singh. Contextualizing commitment protocols. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1345–1352, 2006a.

Amit K. Chopra and Munindar P. Singh. Producing compliant interactions: Conformance, coverage, and interoperability. In *Declarative Agent Languages and Technologies IV:*

Selected, Revised, and Invited Papers, volume 4327 of *LNAI*, pages 1–15, Heidelberg, 2006b. Springer.

Amit K. Chopra and Munindar P. Singh. Interoperation in protocol enactment. In *Declarative Agent Languages and Technologies V: Selected, Revised, and Invited Papers*, LNAI, Heidelberg, 2007. Springer.

William Cook and Jayadev Misra. Computation orchestration: A basis for wide-area computing. *Software and Systems Modeling*, 6(1):83–110, March 2007.

R. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Modeling agent conversations with colored petri nets. In *Working Notes of the Workshop on Specifying and Implementing Conversation Policies*, pages 59–66, Seattle, Washington, May 1999.

Francisco Curbera, Rania Khalaf, Nirmal Mukhi, Stefan Tai, and Sanjiva Weerawarana. The next step in web services. *Communications of the ACM*, 46(10):29–34, 2003.

Luca de Alfaro and Thomas A. Henzinger. Interface automata. In *Proceedings of the Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-9)*, pages 109–120, 2001.

Nirmit Desai and Munindar P. Singh. On the enactability of business protocols. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pages 1126–1131, Menlo Park, July 2008. AAAI Press.

Nirmit Desai and Munindar P. Singh. A modular action description language for protocol composition. In *Proceedings of the 22nd Conference on Artificial Intelligence*, pages 962–967, 2007.

Nirmit Desai, Ashok U. Mallya, Amit K. Chopra, and Munindar P. Singh. Interaction protocols as design abstractions for business processes. *IEEE Transactions on Software Engineering*, 31(12):1015–1027, December 2005.

Nirmit Desai, Amit K. Chopra, Matthew Arrott, Bill Specht, and Munindar P. Singh. Engineering foreign exchange processes via commitment protocols. In *Proceedings of the 4th IEEE International Conference on Services Computing*, pages 514–521, Los Alamitos, 2007a. IEEE Computer Society Press.

Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Representing and reasoning about commitments in business processes. In *Proceedings of the 22nd Conference on Artificial Intelligence*, pages 1328–1333, 2007b.

Frank Dignum, Rogier M. van Eijk, and Roberto Flores, editors. *Agent Communication II*, volume 3859 of *LNCS*. Springer, 2006.

Hywel R. Dunn-Davies, Jim Cunningham, and Shamimabi Paurobally. Propositional statecharts for agent interaction protocols. *Electronic Notes in Theoretical Computer Science*, 134:55–75, 2005.

ebBP. Electronic business extensible markup language business process specification schema v2.0.4, December 2006. docs.oasis-open.org/ebxml-bp/2.0.4/OS/.

Ulrich Endriss, Nicolas Maudet, Fariba Sadri, and Francesca Toni. Protocol conformance for logic-based agents. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 679–684, 2003.

Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. KQML as an agent communication language. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 456–463. ACM Press, 1994.

Roberto A. Flores, Philippe Pasquier, and Brahim Chaib-draa. Conversational semantics with social commitments. In Rogier M. van Eijk, Marc-Philippe Huget, and Frank Dignum, editors, *Agent Communication*, volume 3396 of *Lecture Notes in Computer Science*, pages 18–32. Springer, 2004. ISBN 3-540-25015-8.

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

Nicoletta Fornara and Marco Colombetti. Operational specification of a commitment-based agent communication language. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 535–542. ACM Press, July 2002.

Cédric Fournet, C. A. R. Hoare, Sriram K. Rajamani, and Jakob Rehof. Stuck-free conformance. In *Proceedings of the 16th International Conference on Computer Aided Verification (CAV)*, volume 3114 of *LNCS*, pages 242–254. Springer, 2004.

Xiang Fu, Tevfik Bultan, and Jianwen Su. Conversation protocols: a formalism for specification and verification of reactive electronic services. *Theoretical Computer Science*, 328 (1-2):19–37, 2004.

Dimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–152, April 1995.

Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1-2):49–104, 2004.

Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, October 1996.

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

Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0321200683.

Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Signature Series. Addison-Wesley, Boston, 2004.

Raymond Hu, Nobuko Yoshida, and Kohei Honda. Session-based distributed programming in java. In *22nd European Conference on Object-Oriented Programming*, pages 516–541, 2008.

Marc-Philippe Huget and James Odell. Representing agent interaction protocols with agent UML. In *Agent-Oriented Software Engineering V*, volume 3382 of *LNCS*, pages 16–30. Springer, 2005.

Michael Huhns and David M. Bridgeland. Multiagent truth maintenance. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1437–1445, 1991.

N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *Knowledge Engineering Review*, 2(3):223–250, 1993.

Santhosh Kumaran. Model-driven enterprise. In *Proceedings of the Global Enterprise Application Integration Summit (EAI)*, pages 166–180, 2004.

Ashok U. Mallya and Munindar P. Singh. An algebra for commitment protocols. *Journal of Autonomous Agents and Multi-Agent Systems*, 14(2):143–163, April 2007.

David Martin, Mark Burstein, Drew Mcdermott, Sheila Mcilraith, Massimo Paolucci, Katia Sycara, Deborah L. Mcguinness, Evren Sirin, and Naveen Srinivasan. Bringing semantics to web services with owl-s. *World Wide Web*, 10(3):243–277, September 2007.

Peter McBurney and Simon Parsons. Posit spaces: a performative model of e-commerce. In *Proceedings of the 2nd International Joint Conference on Autonomous agents and Multiagent Systems*, pages 624–631, 2003.

Carlos Molina-Jimenez, Santosh Shrivastava, and Nick Cook. Implementing business conversations with consistency guarantees using message-oriented middleware. In *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*, pages 51–62, 2007.

Timothy J. Norman and Chris Reed. Delegation and responsibility. In *ATAL '00: Proceedings of the 7th International Workshop on Intelligent Agents VII. Agent Theories Architectures and Languages*, pages 136–149, 2001.

OASIS. Oasis web services coordination version 1.1, July 2007. http://docs.oasis-open.org/ws-tx/wscoor/2006/06.

David Lorge Parnas. Information distribution aspects of design methodology. In *Proceedings of the International Federation for Information Processing Congress*, volume TA-3, pages 26–30, Amsterdam, 1971. North Holland.

Shamimabi Paurobally, Jim Cunningham, and Nicholas R. Jennings. Ensuring consistency in the joint beliefs of interacting agents. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 662–669, 2003.

Henry Prakken. Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation*, 15(6):1009–1040, 2005.

RosettaNet. Home page, 1998. www.rosettanet.org.

Michael Rovatsos. Dynamic semantics for agent communication languages. In *Proceedings of the 6th international Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–8, 2007.

John R. Searle. *The Construction of Social Reality*. Free Press, New York, 1995.

Mary Shaw and David Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice-Hall, Upper Saddle River, NJ, 1996.

Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

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

Munindar P. Singh. 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)*, pages 907–914, New York, July 2003. ACM Press.

Munindar P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113, 1999.

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

Munindar P. Singh, Amit K. Chopra, Nirmit Desai, and Ashok U. Mallya. Protocols for processes: Programming in the large for open systems. *ACM SIGPLAN Notices*, 39(12): 73–83, December 2004.

Munindar P. Singh, Amit K. Chopra, and Nirmit Desai. Commitment-based SOA. TR 2007-19, North Carolina State University, January 2008.

Howard Smith and Peter Fingar. *Business Process Management: The Third Wave*. Megan-Kiffer Press, Tampa, 2002.

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

UCC-Sales. Uniform commercial code - Article 2. http://www.law.cornell.edu/ucc/2/overview.html.

Wil M. P. van der Aalst and Maja Pesic. DecSerFlow: Towards a truly declarative service flow language. In *Proceedings of the 3rd International Workshop on Web Services and Formal Methods*, volume 4184 of *LNCS*, pages 1–23. Springer, 2006.

Wil M. P. van der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed Parallel Databases*, 14(1):5–51, 2003.

W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, N. Russell, H.M.W. Verbeek, and P. Wohed. Life after bpel? In *Formal Techniques for Computer Systems and Business Processes*, volume 3670 of *LNCS*, pages 35–50. Springer, 2005.

Mahadevan Venkatraman and Munindar P. Singh. Verifying compliance with commitment protocols: Enabling open Web-based multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 2(3):217–236, September 1999.

Guijun Wang and S. Miller. Intelligent aggregation of purchase orders in e-procurement. In *Proceedings of the Ninth International Enterprise Distributed Object Computing Conference*, pages 27–36, 2005.

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

Michael Winikoff, Wei Liu, and James Harland. Enhancing commitment machines. In *Proceedings of the 2nd International Workshop on Declarative Agent Languages and Technologies (DALT)*, volume 3476 of *LNAI*, pages 198–220, Berlin, 2005. Springer-Verlag.

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

Daniel M. Yellin and Robert E. Strom. Protocol specifications and component adaptors. *ACM Transactions on Programming Languages and Systems*, 19(2):292–333, 1997.

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. ACM Press, July 2002.