Choice and Interoperation in Protocol Enactment*

Amit K. Chopra
Department of Computer Science
North Carolina State University
akchopra@ncsu.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
singh@ncsu.edu

ABSTRACT

Protocols describe interactions among agents and thus underlie the engineering of multiagent systems. However, protocols are enacted by agents in physical systems. In particular, considerations of communication models and how distributed agents are able to make compatible choices would greatly affect whether a protocol may in fact be enacted successfully. The objective of this paper is to study the conceptual underpinnings of protocol enactment in multiagent systems. It seeks to characterize the operationalization of agents so as to determine whether and when agents may be interoperable.

1. INTRODUCTION

Flexibility in protocol modeling and adoption of roles poses great challenges for the enactment of protocols by autonomous agents. Each agent would typically need go beyond the "letter" of any specific protocol in which it participates. Some such extensions would be safe, yet other innocuous extensions could cause failures in interoperation (most often manifested as deadlocks). For example, while participating in payment, an agent may also participate in shipping, thus going beyond what is specified in payment. But if the agent were to request an additional quote during payment or even to send a reminder, it may fail to interoperate with its payment partner: e.g., because the partner wouldn't give an extra quote or didn't expect a reminder.

We define interoperability on the basis of a simple notion of blocking: if an agent is blocked indefinitely waiting for a message from some other agent to arrive, and is thus unable to make any progress, then the agents are noninteroperable. The definition is inspired from the observation that an agent's goal in an interaction with another agent is to reach an acceptable final state. If the agents under consideration can always reach acceptable termination states by communicating with each other, then they are interoperable.

A communication model sets the physical environment for communication between agents. The parameters of this model include whether communication is synchronous or asynchronous, the number of channels to use, the sizes of the buffers, and the buffer access mechanism. One cannot simply examine a pair of agents in isolation and decide whether they are interoperable; the agents must be analyzed in light of the communication model in force. Agents that are interoperable in one model may be noninteroperable in another. To analyze interoperability, we build communication models based on the notion of *causal enablement*. Causal enablement is a basic building block that identifies the possible nonblocking actions given the current global state of the interaction. Different models of causal enablement correspond to different communication models. This paper considers causal enablement for a model where communication is asynchronous, only one channel exists between agents, and the associated buffer is unbounded RAM.

Another aspect of the operationalization of agents that affects interoperability is choice compatibility. At various points during its interactions with other agents, an agent may have a choice in its actions. The action it chooses may not be compatible with the action chosen by the agent it is interacting with. Such agents may be termed noninteroperable by a naive interoperability test despite the fact that such agents would be deemed interoperable in practice. We outline a relaxed interoperability criterion, which depends on identifying potentially risky choice states.

2. VERIFYING INTEROPERABILITY

We represent the computations of agents as transition systems. A transition system is a graph with states as vertices and actions as edges. A transition is a triple $\langle s, e, s' \rangle$ where s and s' are states, and e is a set of actions. In addition, an initial state and final states are marked. We model two types of actions in agents: send and receive. A send is indicated by $!p$, and a receive is indicated by $?p$, where $p, q, \ldots$ range over messages $x, y, \ldots$. The sender and recipient of a message are not made explicit, as we consider only two-party interactions here. Furthermore, we consider only single-threaded agents; hence, in an agent's transition system, each transition is labeled with one action only.

The interoperability of two agents depends upon the computations that they can jointly generate. These computations are obtained by taking the product of the individual agents' transition systems. The agents may act one by one or in true concurrency. When the agents act one by one, the transitions in the product are labeled with their respective actions. When the agents act concurrently, the transitions are labeled by a pair of actions, one from each agent. Figures 1–7 each contain three transition systems: one for agent α (identified by states labeled with one digit), one for agent β (identified by states labeled with one digit followed by an apostrophe as in $0'$), and their product (identified by states that contain states labeled with two digits). The dashed edges are also part of the product; their significance is explained later.

We say that an action is *causally enabled* in a state of the tran-

*This research was partially supported by the National Science Foundation under grant DST-0139037.

AAMAS'07, May 14–18 2007, Honolulu, Hawaii, USA.

978-81-904262-7-5 (RPS) ©2007 IFAAMAS

sition system if attempting the action in that state completes successfully. A transition is causally enabled if all the actions in it are causally enabled. An attempt to do a send action is always causally enabled under the communication model described above. An attempt to do a receive action, however, completes successfully only if the corresponding send action has happened prior to it, or in concurrency with it. If a receive action is not causally enabled in a state and it is attempted, then we say that the agent is *blocked*.

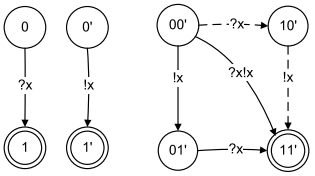

Figure 1: Simple agents (interoperable)

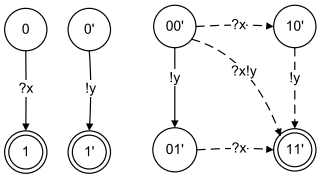

Figure 2: Simple blocking agents (noninteroperable)

A path in a transition system is a series of transitions from the initial state to a final state. A path is said to be *realizable* if all the transitions in it are causally enabled. Figure 1 shows two simple agents and their resulting product. Here, paths $\langle 00', \{!x\}, 01', \{?x\}, 11' \rangle$ and $\langle 00', \{!x?x\}, 11' \rangle$ are both realizable. (In all the figures, the realizable paths consist only of solid edges, whereas the nonrealizable paths have dashed edges.) A path is said to be *pathological* if it is not realizable, but a realizable path exists between its initial and final states. Pathological paths have no bearing on the interoperability of agents; they may be thought of as spurious permutations of a realizable path. In Figure 1, the path $\langle 00', \{?x\}, 10', \{!x\}, 11' \rangle$ —an impossible interleaving of the transitions in the realizable path $\langle 00', \{!x\}, 01', \{?x\}, 11' \rangle$—is pathological. A path is said to be *blocking* if it is not realizable and there exists no realizable path between its initial and final states. In Figure 1, there are no blocking paths because the only final state $11'$ is reachable by a realizable path. However, Figure 2 shows two agents, one of which is attempting to receive a message (x) that the other never sends. Hence all of the three paths in their product are blocking. We say that the final state of a blocking path is *noncausal*.

DEFINITION 1. *Two agents α and β are interoperable if there exists no blocking path in their product, or in other words, no final state is noncausal.*

Thus, agents in Figure 1 are interoperable, whereas agents in Figure 2 are noninteroperable because state $11'$ is noncausal. Figure 3 shows two agents with a symmetric choice: one can send x or y, and the other can receive x or y. If one agent sends x and the other attempts to receive y, then the receive would block. As expected, the agents are noninteroperable because state $12'$ in the product is noncausal.

In Figure 4, the right agent sends y (contrast with the right agent in Figure 3, which sends x or y). If the left agent attempts to receive x, then the receive would block. As expected, these agents are noninteroperable because state $11'$ in the product, which is a final state, is noncausal.

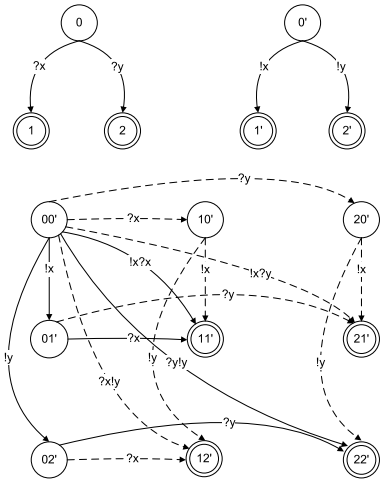

Figure 3: Agents with a symmetric choice (noninteroperable)

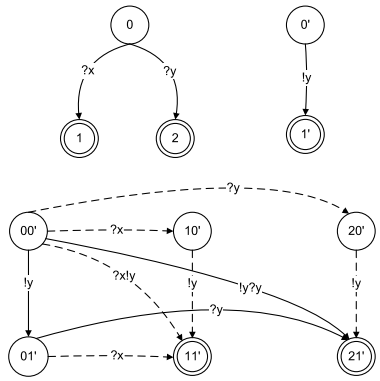

Figure 4: Agents with limited send choice (noninteroperable)

The agents in Figure 5 are also a variation on the agents in Figure 3. This time the receiving agent has no choice. If the other agent sends x, then the attempt to receive y would block. As expected, the agents are noninteroperable because state $11'$ in the product, which is a final state, is noncausal.

In Figure 6, one agent receives messages in an order opposite to the order that they were sent in. These agents are nevertheless interoperable. This is as expected with random access buffers. The agents in Figure 7 are noninteroperable because state $21'$ in the product—a final state—is noncausal. This is as expected because the agents can deadlock if they both receive first.

As shown above, the agents in Figure 3 (symmetric choice) are noninteroperable. In Figure 3, state $02'$ is one state where a choice exists between what to receive—if the agent chooses to do $?x$, the agent blocks; if the agent chooses $?y$ it does not block. State $02'$ is thus a *risky* state. A risky state is one in which at least one message can be received, and in which attempting to receive some message will block. It is risky for the agent because although a nonblock-

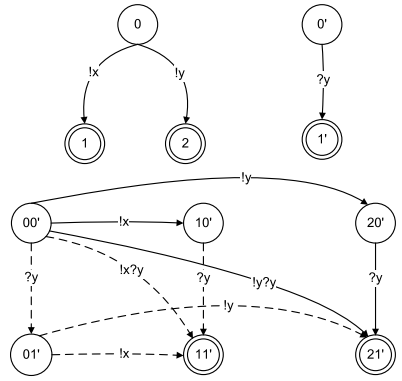

Figure 5: Agents with no receive choice (noninteroperable)

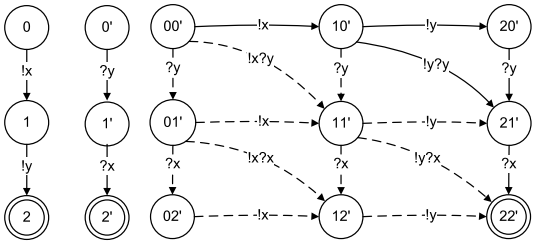

Figure 6: Agents with out of order receives (interoperable)

ing receive exists out the state, the agent has no way of knowing which one. States $00'$ and $01'$ are other risky states in Figure 3. If there were a way to *disable* the blocking receives in a risky state (i.e., to ensure that such a receive would never be attempted), then the agents in Figure 3 would be interoperable. Under a similar setting, the agents in Figure 4, which were previously noninteroperable would become interoperable—in this case, states $00'$ and $01'$ are risky states.

In practice, there are many pairs of agents such as those in Figures 3 and 4 that we expect to be interoperable. For example, customer agents in a purchase protocol may have a choice in accepting or rejecting a quote sent by the merchant. Clearly, such agents should not be determined noninteroperable based only on that. The following outlines a way of handling receive choices in agents.

In a risky state, we want the agent to magically do a receive that does not block. The trick to handling receive choices is to have the agent not attempt to receive specific messages; instead, an agent should attempt to read its buffer for whichever message is available from among the choices it has. We give each agent an angel. An agent's angel knows if some message from among the agent's choices is available to be received by the agent. If a message is available, it delivers that message to the agent. Thus the angel makes the choice for the agent. However, if no message is available, the agent blocks.

3. DISCUSSION

Interoperation has bearing on the ability of agents to comply with the protocols in which they participate. They might each individually be designed to be conformant with a protocol but may fail to comply with the protocol only because they cannot interoperate with their partners on the field. Chopra and Singh [2] formalize

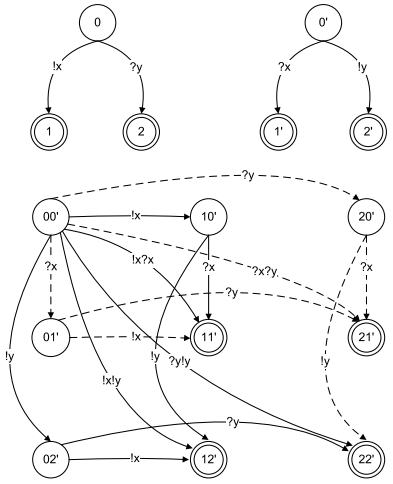

Figure 7: Agents that deadlock (noninteroperable)

interoperability and conformance, and show that they are orthogonal properties. Baldoni *et al.* [1] and Endriss *et al.* [3] propose alternative notions of interoperability that are closely tied to conformance. Consequently, Baldoni *et al.* and Endriss *et al.* offer definitions of interoperability that are more restrictive than the one proposed here.

Kazhamiakin *et al.* [4] construct a hierarchy of communication models depending on factors such as synchrony and buffers among other, and use this framework to find the best fit communication model for a given service composition such that the cost of further verification is cheapest. However, their method assumes that the services are composable under some model; they present no algorithm that decides composability.

Besides generalizing the claims in the paper to general multiparty interactions, an important direction of work is to study the properties of goal-based interoperability as is proposed here.

4. REFERENCES

[1] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. Verification of protocol conformance and agent interoperability. In *6th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA VI)*, pages 265–283, 2005.

[2] A. K. Chopra and M. P. Singh. Protocol compliant interactions: Conformance, coverage, and interoperability. In *Declarative Agent Languages and Technologies IV: Selected, Revised, and Invited Papers*, *LNAI* 4327. Springer, 2006.

[3] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Protocol conformance for logic-based agents. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 679–684, 2003.

[4] R. Kazhamiakin, M. Pistore, and L. Santuari. Analysis of communication models in web service compositions. In *Proceedings of the 15th International Conference on World Wide Web*, pages 267–276, 2006.