Commitments and Causality for Multiagent Design

Feng Wan
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535, USA
fwan@eos.ncsu.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535, USA
singh@ncsu.edu

ABSTRACT
This paper unifies two recent strands of research in multiagent system design. One, commitments are widely recognized as capturing important aspects of interactions among agents, but current approaches tend to emphasize individual commitments and typically restrict themselves to interactions between pairs of agents. Two, methodologies for multiagent system design consider protocols and coordination requirements, but do not seriously accommodate commitments. This paper proposes a methodology to infer commitments from an example conversation among several parties. Based on the conversation, we build a commitment causality diagram indicating the causal relations among the commitments. Using this diagram, we generate behavior models for each role. We show that the models produced successfully capture commitment-level protocols and allow flexible implementation of non-commitment communications provided the causal relations are preserved.

Categories and Subject Descriptors
1.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence

General Terms
Design

Keywords
Commitments; roles; conversation analysis; Dooley graphs

1. INTRODUCTION
Commitments (also known as social commitments) are widely recognized as a key representation for the interactions in a multiagent system. This is because commitments enable us to model and analyze the behavior of autonomous agents, especially in settings such as supply chain integration and Web service composition.

Commitments are a key element of the semantics of agent communications [Singh, 2000a]. Recent work operationalizes commitments. Fornara and Colombetti develop an operational semantics that models the lifecycle of a commitment [2002]. Economou et al. show how deontic states and commitments can be detected from agents’ finite state machines and how communications rely upon the protocols as executed by each agent [2001]. Yolum and Singh show how protocols among agents can be encoded as commitment machines and automatically executed [2002].

Another important theme is the design of multiagent systems through a deep analysis of their desired interactions. Dooley graphs separate agent interactions into courses of conversations [Parunak, 1996]. Each independent conversation reflects a fragment of agents’ models and shows how agents behave in different stages of their interactions. Singh showed how to generate coordination requirements based on the agent skeletons created from a Dooley graph [2000b].

Based on the foregoing, the main idea of this paper is to analyze agent interactions in terms of causally linked commitments, and to develop a methodology (with significant algorithmic components) using which commitment-based agents can be inferred. The advantage of doing so is to produce richly structured multiagent systems whose members interact flexibly.

Agent interactions can be naturally understood in terms of the commitments they make, fulfill, cancel, or modify. As our running example, consider a travel planning scenario. A customer (or passenger $P$) calls his travel agent ($T$) to book a trip. He makes a commitment that if $T$ books the trip for him, he will pay for the trip as well as any processing costs. Upon receiving the order, $T$ sends requests to airline ($A$), hotel ($H$), and car rental ($R$) agents to reserve air tickets, hotels, and cars, respectively. $T$ also makes commitments that if $A$, $H$, and $R$ accept his requests and if $P$ purchases the trip, then he will pay them. If $A$ finds an available flight, he will make a commitment to reserve it. So will $H$ and $R$, if they have available spots. Eventually, if all goes well, $T$ will make a commitment to $P$ to confirm the trip. However, $P$ may cancel all or part of the trip. For instance, $P$ may cancel the car rental if he will get a ride with a colleague. In this case, $P$ updates his request; that is, he updates his original commitment. This update may cause $T$ to update and cancel some of his commitments.

From this example, we can see that the essential states of the protocol correspond to the creation, update, cancellation, or fulfillment of each commitment. These states form our domain-specific protocol requirements; transitions among them are the main interactions. Therefore, if we can identify the commitments and capture their causal relations, then we can potentially generate agent models and build a flexible and robust multiagent system.

This paper is organized as follows. Section 2 provides a background on Dooley graphs with some necessary enhancements. Section 3 shows how to derive commitments. Section 4 captures the causal relations among the commitments and generates agent behavior models. Section 5 discusses the relevant literature.
2. AGENT INTERACTIONS

The starting point for design with Dooley graphs is to analyze an interaction by classifying the messages exchanged (communicative acts) in it and tagging the key relationships among the acts. Parunak proposed the acts Request, Refuse, Commit, Question, Inform, ACT, to which we add Cancel. ACT refers to an action that is external to the commitments, e.g., to discharge a particular commitment. Cancel simply enables a requester or a requestee to cancel his request or commitment, respectively. We also treat Request as a conditional commitment (defined in Section 3) made by a requester.

Parunak defined four relationships among pairs of utterances, namely, respond, reply, resolve, and complete. We add a fifth relationship, termed update. The following is the complete set of acts and relationships identified. Notice the Commit utterances, the above five relations help identify conditions mutual and multiparty scenarios [Singh, 1999].

Although unconditional commitments can be directly obtained from Commit utterances, the above five relations help identify conditional commitments (Requests) and causal relations among them.

We can now capture the necessary details in our trip planning example. Table 1 shows an interaction with the key communicative acts and relationships identified. Notice the $H_1$ and $H_2$ are two hotels; $H_1$ is contacted when $H_2$ cancels. Similarly, $A_1$ and $A_2$ are two airlines. Figure 1 illustrates the corresponding Dooley graph based on the algorithm given by Singh [2000b]. Parunak modularizes conversations to reuse agent modules [1996]. Singh focuses on deriving individual agent models from the relationship of the conversations [2000b]. Huhns et al. incorporate exception handling into the characters involved in each agent model [2002].

However, the above approaches concentrate on low-level interactions, such as orderings of events. They do not study how high-level abstractions, such as commitments, can be induced. This paper demonstrates how commitments and relations among them can be derived from Dooley graphs and what the advantages are of looking at the models at commitment level.

3. DERIVING COMMITMENTS

The following definitions are important for our approach.

- **Characters**. These are the vertices in a Dooley graph. They represent the individual entities at different stages of the modeled interaction. In our example, the list of characters is $\{P_1 – P_3, T_1 – T_8, A_1 – A_2, H_1 – H_2, R\}$.

- **Conversation**. A sequence of utterances between two characters derived from a Dooley Graph. A character can only participate in one conversation. In Figure 1, the set of conversations is $\{\chi_1 = \{1,10\}, \chi_2 = \{2,4\}, \chi_3 = \{5,6,21\}, \chi_4 = \{3,8,14\}, \chi_5 = \{7,9,12\}, \chi_6 = \{11,13\}, \chi_7 = \{15,16,19,20\}, \chi_8 = \{17,18\}\}$.

- **Conversation initiator**. The sender of the first utterance in a conversation; e.g., the conversation initiator of $\chi_2$ is $T_2$.

- **Causality**. If the send or receive of an $u_k$ precedes $u_j$ then $u_k$ is a potential cause of $u_j$. For a given conversation, the causes are annotated via the various relations among utterances. The internal conditions that enable the causes are not explicitly specified to preserve the autonomy and heterogeneity of the participants.

- **Context-related conversations**. Let utterances $u_k$ and $u_j$ occur in conversations $\chi_k$ and $\chi_j$, respectively. If $u_k$ updates $u_j$ then $\chi_k$ is context-related to $\chi_j$. For example, $\chi_1$ and $\chi_6$ are context-related conversations.

- **Role**. The abstraction of capabilities used by characters who deal with same type of transactions or are involved in context related conversations. In Figure 1, the characters inside a dotted circle belong to same role.

- **Agent**. A concrete party who can play one or more roles. Our example has agents $\{P, T, A_1, A_2, H_1, H_2, R\}$. Notice that agent $T$ plays 4 roles and has 8 characters, and $A_1$ and $A_2$ play the same role, which has characters $A_1$ and $A_2$.

- **Commitment**. An obligation from a debtor to a creditor about a particular condition. For debtor $x$, creditor $y$, and condition $p$, the relevant commitment is notated $C(x,y,p)$. Modeling commitments as directed from debtor to creditor simplifies them. But they support seven operations and combine to capture mutual and multiparty scenarios [Singh, 1999].

![Figure 1: Dooley graph for trip planning](image-url)
Table 1: Possible annotated interactions in the trip planning example

Unconditional commitment. A commitment whose condition is a simple proposition.

Conditional commitment. A commitment of the form $C(x, y, e \rightarrow p)$, where $e$ is a condition (possibly interpreted as an event) and $p$ is a condition to be brought about (possibly interpreted as an action). $p$ is activated when $e$ becomes true.

Our communicative acts map to the following commitment operations: create, update, discharge, and cancel.

- **Request (without Update).** Create $C(x, y, e \rightarrow p)$. A conversation initiator usually requests a particular information or service. The antecedent $e$ corresponds to a condition potentially satisfied by the requestee, e.g., a commitment to do something. The consequent $p$ corresponds to what the requestor will do if the requestee satisfies $e$. Therefore, this action forms a conditional commitment.

- **Request and Update.** Update $C(x, y, e \rightarrow p)$. This is a commitment update sent by a request creator. Either $e$ or $p$ could be changed, but the modified commitment still follows the original organizational rules.

- **Commit and Resolve.** Create $C(x, y, p)$. If a Commit resolves an utterance, then it creates a commitment for the first time following the applicable rules. This may caused by a request sent from a conversation initiator and is also decided by the willingness of the commitment creator.

- **Commit and Update.** Update $C(x, y, p)$. If a Commit updates an utterance, then it updates its original commitment. It is equivalent to cancel the previous commitment and create a new one but still following the original rules.

- **Act and Complete.** Discharge $C(x, y, e \rightarrow p)$ or $C(x, y, p)$. If an act (i.e., a non-communicative act) completes an utterance, then the action performer fulfills and discharges the commitment made in that utterance. Since a commitment need not be fulfilled in one message, several (Act and Complete) utterances may exist to complete the same utterance.

- **Act and Resolve.** Create and discharge $C(x, y, p)$. If a non-communicative act resolves an utterance, then the action performer creates and fulfills a commitment at same time.

- **Cancel and Complete.** Cancel $C(x, y, e \rightarrow p)$ or $C(x, y, p)$, if a Cancel completes an utterance, then it cancels a conditional commitment (Request) or an unconditional commitment (Commit) made in that utterance.

From the mappings, we can prove that commitments are created within the boundaries of conversations because the commitments are created by utterances and, as a consequence, both debtor and creditor are characters in a same conversation. Algorithm 1 below generates a complete list of commitments.

The algorithm creates a conditional commitment for a Request message, e.g., $C(P, T, e_1 \rightarrow p_1)$ for utterance $u_1$. It creates an unconditional commitment for a Resolve message, e.g., $C(A_2, T, Confirm)$ for utterance $u_k$. To decide the antecedent and consequent in a conditional commitment, the algorithm looks for an Act and Complete message $u_k$ that completes a Request utterance $u_{kj}$. If there is one, then the Act decides the consequent. For any utterance $u_k$ that $u_k$ responds to, the corresponding commitments or acts of $u_k$ constructs the antecedent.

By executing the algorithm on Table 1, we obtain the following list of commitments. Here TBD indicates to be decided. All conditions of the form $true \rightarrow q$ are simplified to $q$.

- $C_1 = C(P, T, C(T, P, SendItinerary) \rightarrow Pay(P, T))$
- $C_2 = C(T, A_1, TBD)$
- $C_3 = C(T, A_2, C(A_2, T, Confirm) \land Pay(P, T) \rightarrow Pay(T, A_2))$
- $C_4 = C(T, H_1, TBD)$

<table>
<thead>
<tr>
<th>#</th>
<th>S</th>
<th>R</th>
<th>Act Type</th>
<th>Utterance</th>
<th>Respond to</th>
<th>Reply to</th>
<th>Resolve</th>
<th>Complete</th>
<th>Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P$</td>
<td>$T$</td>
<td>REQUEST</td>
<td>Book trip</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$T$</td>
<td>$A_1$</td>
<td>REQUEST</td>
<td>Buy ticket</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$T$</td>
<td>$H_1$</td>
<td>REQUEST</td>
<td>Reserve hotel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$A_1$</td>
<td>$T$</td>
<td>REFUSE</td>
<td>Not Available</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$T$</td>
<td>$A_2$</td>
<td>REQUEST</td>
<td>Buy Ticket</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$A_2$</td>
<td>$T$</td>
<td>COMMIT</td>
<td>Confirm Ticket</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$T$</td>
<td>$R$</td>
<td>REQUEST</td>
<td>Rent car</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$H_1$</td>
<td>$T$</td>
<td>COMMIT</td>
<td>Confirm Hotel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>$R$</td>
<td>$T$</td>
<td>COMMIT</td>
<td>Confirm Car</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>$T$</td>
<td>$P$</td>
<td>COMMIT</td>
<td>Send Itinerary</td>
<td>6, 8, 9</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>$P$</td>
<td>$T$</td>
<td>REQUEST</td>
<td>Cancel Car from the Itinerary</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>$T$</td>
<td>$R$</td>
<td>CANCEL</td>
<td>Cancel Car</td>
<td>11</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>$T$</td>
<td>$P$</td>
<td>COMMIT</td>
<td>Send Revised Itinerary</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>$H_1$</td>
<td>$T$</td>
<td>CANCEL</td>
<td>Cancel Hotel</td>
<td>11</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>$T$</td>
<td>$P$</td>
<td>QUESTION</td>
<td>Alternate Hotel?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>$T$</td>
<td>$T$</td>
<td>INFORM</td>
<td>Yes</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>$T$</td>
<td>$H_2$</td>
<td>REQUEST</td>
<td>Reserve Hotel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>$H_2$</td>
<td>$T$</td>
<td>COMMIT</td>
<td>Confirm Hotel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>$P$</td>
<td>$P$</td>
<td>COMMIT</td>
<td>Send Revised Itinerary</td>
<td>18</td>
<td>16</td>
<td>16</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>$T$</td>
<td>$T$</td>
<td>ACT</td>
<td>Pay for the trip</td>
<td></td>
<td></td>
<td>19</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>$T$</td>
<td>$A_2$</td>
<td>ACT</td>
<td>Pay for the ticket</td>
<td>6, 20</td>
<td>6</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CommitList = \{\}
for each \(u_k\) do
switch \(u_k\) do
\text{case Request:}
Add \(C(S_i, R_j, e_i \rightarrow p_i)\) to CommitList, where \(e_i = \text{true}\) and \(p_i = \text{ToBeDecided}\).
\text{case (Commit and Resolve) or (Act and Resolve):}
Add \(C(S_i, R_j, p_k)\) to CommitList, where \(p_k\) is an action or a condition that \(S_i\) commits to \(R_j\) to perform or keep true.;
\text{case Act and Complete:}
\(u_j\) \leftarrow the utterance that \(u_k\) completes;
if \(u_j\) is a Request then
\(p_j\) \leftarrow the Act;
for each \(u_k\) that \(u_k\) responds to do
switch \(u_k\) do
\text{case Commit + Resolve:}
\(e_j \leftarrow e_j \wedge C(S_k, R_k, p_k)\);
\text{case Act + Resolve:}
\(e_j \leftarrow e_j \wedge \text{the Act};\)
\text{case Update:}
\(u_t\) \leftarrow the original Resolve utterance that \(u_k\) updates following the update chain;
\(e_j \leftarrow e_j \wedge C(S_i, R_i, p_i)\);
\text{case Act + Complete:}
\(e_j \leftarrow e_j \wedge \text{the Act};\)
\end{algorithm}

Algorithm 1: Generate a list of commitments

- \(C_2 = C(T, R, \text{TBD})\)
- \(C_0 = C(A_2, T, \text{Confirm})\)
- \(C_1 = C(T, R, \text{Confirm})\)
- \(C_0 = C(R, T, \text{Confirm})\)
- \(C_0 = C(T, P, \text{SendItinerary})\)
- \(C_1 = C(T, H, \text{TBD})\)
- \(C_11 = C(H_2, T, \text{Confirm})\)

Some conditional commitments have undecided antecedents and consequents, because our initial table does not provide enough information to derive the dependencies. For example, for commitment \(C_2\), because \(A_1\) declines the request, we cannot determine what \(A_2\) will commit to \(T\) if he accepts the request. For commitments \(C_5, C_8, \text{and } C_{10}\), neither the antecedent nor the consequent of \(T\) can be determined, because there is no \textit{Act and Complete} utterance sent by \(T\) in response to the commitments made by other parties. However, the travel domain may have a business rule or policy that cancellation charges may be owed, e.g., if a passenger is a no show. Any such domain policies can be added during the final design.

Many commitment instances are derived from the conversation table. Some commitments update the others and some replace the others, but they essentially relate to same debtors and creditors and deal with same transactions. We treat all these related commitments as the instances of the same commitment class. They are merely a result of operation performed on the same class. Next, we give the definition of a commitment class. Two commitments \(C(x_1, y_1, p_1)\) and \(C(x_2, y_2, p_2)\) belong to same class if and only if they satisfy the following requirements:

1. \(\text{Role}(x_1) = \text{Role}(x_2)\) and \(\text{Role}(y_1) = \text{Role}(y_2)\)
2. \(p_1\) and \(p_2\) deal with same transaction, goods, or information (possibly \(p_1 = p_2\)).

Based on this definition, we can put commitment \(C_2\) and \(C_3\) into same class, because the only difference between them is that different instances of the same role \(A\) (\(A_1\) and \(A_2\)) are involved. Likewise, \(C_4\) and \(C_{10}\) are classified together. For each class, we use the commitment with the lowest subscript as the representative of the class but with bold font. Therefore, the above two classes are named \(C_3\) and \(C_{10}\), respectively.

For a commitment class, any characters involved in its instance commitments are replaced by their corresponding roles. For example, in commitment class \(C_2\), character \(A_1\) and \(A_2\) are both replaced by \(A\), and in class \(C_4\), \(H_1\) and \(H_2\) are replaced by \(H\).

If a commitment class is a conditional commitment, then the antecedent of the class is the conjunction of the antecedents of the constituting commitments, and the consequent is the list of consequents of those commitments. For example, \(C_3 = C(T, A, (C(A, T, \text{Confirm}) \wedge \text{Pay}(T, T)) \rightarrow \text{Pay}(T, A))\) and \(C_4 = C(T, H, \text{TBD})\). This is appropriate because the commitments in a class involve the same transactions.

### 4. Deriving Causal Relations

Causal relations among commitments are crucial to understanding the chain of commitment operations, because they drive an interaction along significant states where the real transactions of domain value occur. A commitment causality diagram (CCD) is a graph showing potential causality between each pair of commitment operations. A CCD highlights the important stages within the information flows and hides details of the interaction protocols that can vary depending on the actual implementation. From a designer’s standpoint, a CCD reflects the high-level business logic that specifies what agreements should be achieved. From a CCD, we can infer the agent conversations needed to achieve and modify the commitments underlying these agreements. Once the commitments and their relations are identified, designers can always choose an optimal conversation that flows between any related pair of commitment operations.

Figure 2 shows a CCD derived for our running example. Each node consists of five elements, namely, the commitment class identifier and its associated four operations: create (Crt), update (Upd), discharge (Dcg), and cancel (Cnl). If an operation of a commitment is causally related to another operation, then there is a directed edge from the causing operation to the caused operation. For example, the creation of commitment \(C_1\) causes the creation of commitment \(C_2\), then there is an edge from Crt(\(C_1\)) to Crt(\(C_2\)).

Algorithm 2 generates a commitment causality diagram. This algorithm scans through each commitment operation and finds its immediate causally related operations. It examines each Responds path started from the original operation, since one operation can cause the occurrence of multiple other operations. The very first operation in each path must be the only operation directly caused along that path. For example, Figure 3 shows the Responds path of utterance \(u_1\), wherein \(u_2\), \(u_3\), and \(u_7\) are directly caused by \(u_1\), but
u_5 and u_6 are not. Notice that since multiple paths are considered, the occurrence of the original operation does not entail the occurrence of the first operation of any path. A subtlety in this algorithm is that different causal paths can pass through the same commitment operation (see line 8 of Algorithm 2). This enables the given role to be bound to different agents at run time. Figure 2 illustrates this point. Recall that C_4 deals with the interaction of T with H (derived from H_1 and H_2). The cancellation of the instance of C_7 causes a creation of a new instance of C_4 for H_2, which automatically voids the instance of C_4 for H_1. A dotted edge is added between Cnl(C_7) to Cnl(C_4) to indicate this. The new instance of C_4 then triggers a new instance of C_7. The above scenario is reflected in the edges labeled \( \chi_{8} \) and the dotted lines within the Crt(C_4) and Crt(C_7) blocks.

A CCD illustrates how commitments evolve during interactions. For example, in Figure 2, P’s initial request, which corresponds to the creation of C_1, triggers the creation of C_2, C_3, and C_7. This is a simple request, which follows a travel domain rule that causes T to create three subsequent requests without further negotiation. The creation of the first instance of C_2 leads to the creation of another instance of C_2, because T and A_1 failed to negotiate in conversation X_2. In this case, the first instance of C_2 is implicitly canceled (we add a dotted edge from Crt(C_2) to Cnl(C_2)). The second instance of C_2 leads to the creation of C_3, because A_2 in conversa-
tion $\chi_3$ accepts the request and sends out a confirmation. This case indicates that the creation of $C_2$ may lead to the creation of $C_3$, but if it fails, it will be created again by $T$ without any additional negotiation or consultation. In a real system, this may happen when the first preferred airline is sold out and $T$ tries another acceptable one, if there is any. Because there must be a limit on the number of tries, when there are no more airlines to consider, the cancellation of $C_2$ may lead to the cancellation or update of $C_1$. For example, $P$ may want to cancel the whole trip or change the schedule of the trip.

Let us look at another case in which $P$ updates the trip order by canceling the car rental. This corresponds to the update of $C_1$, which leads to the cancellation of $C_3$ and update of $C_4$. There is no edge between $\text{Cnl}(C_5)$ and $\text{Cnl}(C_6)$. However, to model a travel domain rule that the cancellation of car rental request will cancel any car rental confirmation, we add a dotted edge between those two cancellation operations.

The last case we study is the cancellation of $C_7$. This leads to the creation of another instance of $C_8$ by going through the conversation $\chi_7$; in which case $P$ asks $T$ for an alternative hotel. This case tells us that between each pair of commitment operations, there may be more conversations and roles involved to decide whether the conditions behind the potential causal relation are satisfied at run time.

By analyzing the CCD we can get a view on how commitments are created, updated, and discharged in a normal business flow and how they get canceled because of exceptions or revoked requests. This view is not complete since the original conversation table provides only partial case scenarios. We can either provide more complete agent interactions or try to complete and refine the diagram by hand. The first option is obviously not acceptable, since you have to provide many low-level messages and can be easily confused in trying to capture causal relationships among all the messages. The second option is more appealing since we would already have captured significant transactions and events at the commitment level—the only things remaining would be additional causal edges between some pairs of commitment operations.

### 4.1 Accommodating Commitment Patterns

In previous research, we proposed a small number of commitment patterns so as to streamline the interactions among agents [Xing et al., 2001]. The main patterns are Entertain Request, Notify the Consumer, Entertain Update, Renotify the Consumer, Satisfy the Consumer (Retry), and Resign. These are the behaviors of a single agent. However, by applying them to a CCD, we can state that any operation performed on one commitment needs to be propagated to the operation of another commitment to which the first one is causally related. For example, if the creation of one commitment causes the creation of another one, then updating or canceling the first one should affect the second one too.

In the CCD of Figure 2, there is no out-edge from the cancellation of $C_1$, which corresponds to the request cancellation from $P$. We can add three edges connecting this node to the cancellations of $C_2$, $C_3$, and $C_4$, respectively. If this cancellation happens after the discharge of $C_1$ (in which case $P$ has paid for the flight ticket), then a new conversation may be involved between $\text{Cnl}(C_1)$ and $\text{Cnl}(C_2)$ that will resolve a refund or surcharge.

For any existing edges, we can optimize or reduce the conversations along the edges. For example, the cancellation of hotel ($\text{Cnl}(C_7)$) creates a new $C_4$ through conversation $\chi_9$. Can we eliminate this conversation? For instance, automatically find another hotel within two miles of the old one, so that we can avoid potential delay resulting from an email round trip or a phone call.

Please note that any additional conversations may spawn new commitments. We don’t allow commitment operations on the causality edge between two other commitment operations. Therefore, we need to repeatedly regenerate the CCD whenever new conversations are added.

### 4.2 Generating Agent Models

![Travel agent behavior model statechart](image)

Figure 4: Travel agent behavior model statechart

The last step of our approach is generating agent models. The agent models are represented in commitment centric statecharts and are derived from the commitment causal relation diagram. Each statechart is composed of two parts, dealing with creation and Revision, respectively. The creation phase represents the negotiation of initial agreements and the formation of commitment chains, while
the Revision phase represents the update and cancellation of the agreements and any subsequent effects on the commitment chains. Algorithm 3 generates a statechart for an agent. For reasons of space, the algorithm is not as detailed as the others, but highlights the major steps. Figure 4 is the example statechart for $T$.

1. For each operation $\alpha$ on each commitment created by agent $A$ do
2. Create a new state $s_1$;
3. Create a new transition to $s_1$ and mark it as "Send $\alpha$";
4. For each incoming edge to $\alpha$ do
5. $\beta$ ← The source operation of the edge;
6. Create a new state $s_2$;
7. Create a new transition to $s_2$ and mark it as "Receive $\beta$" or any communication involved on the edge;
8. Construct the "creation" part of the statechart by linking and merging the states and transitions that involve only the creation of commitments;
9. Construct the "revision" part of the statechart by Linking and merging the states and transitions that involve update, cancel, discharge or the second creations of commitments;
10. Connect these two statecharts together by the state where the last commitment is created;
11. Put any concurrent states and transitions into AND states and merge their common transitions;
12. Use AND connector to converge transitions leading to a same state if these transitions are the AND preconditions of the state;

Algorithm 3: Generate statechart for an agent

5. DISCUSSION

Agent-oriented software engineering (AOSE) has been a popular topic for building electronic commerce systems and global supply chains. Recent approaches have considered the purely methodological aspects of AOSE. Our program of research seeks to develop rigorous methodologies for AOSE and for multiagent systems design with the view of supporting AOSE. Our contribution here is on the technical aspects of how to create commitment-based representations.

Parunak originated the idea of using Dooley graph to decompose agent conversation and differentiate courses of stages, so agent interaction can be modularized and reused [1996]. This is a good starting point to scoping characters and conversations which is very helpful to design an agent behavior model. We extend this approach by adding more communicative acts (cancel request) and an additional relation among utterance (update), so agent conversations become more meaningful in terms of the relations with each other and business process requirement can be more accurately captured.

Singh previously found the Dooley graph is capable of deriving skeletons for heterogeneous agents [2000b]. However, the main results dealt with the low-level agent interaction protocols. Our study found that the commitment dependencies are more useful than commitments themselves since the lifecycle of each commitment totally rely on the relations with other commitments and actions. They could be nesting deeply with each other. Any break in the commitment dependency chain could affect all related commitments. Therefore the major advantage of our approach over Singh’s is to generate agent models from commitment relations, but not from the message sequence.

Huhns et al. use Dooley graphs to deal with exception handling in supply chains [2002]. After deriving an agent conversation and an agent model, they add branches to the agent skeletons to capture possible exception. However, the effects of agent internal exceptions on other agents’ behavior are not clearly captured. Our approach generalized exception handling into two commitment operations, cancel and update. Our commitment causal relation diagram shows what are the chains of effects of these operations, so the designer may have generic way to deal with different types of exceptions occurred in the whole system.

Fornara and Colombetti [2002] present an operational semantics of commitments relating to communicative acts. They mapped out each communicative act to a commitment operation and depict a commitment state diagram during a course of conversations. This greatly helps us to discover the protocols of making and fulfill commitments within given conversations and differentiate each substate in the negotiation process. However, like most other commitment-based approaches, this one still deals with commitments between two agents, the relations among commitments are not explored.

Verdicchio and Colombetti recently presented ideas of generalizing commitments relations in supply chain systems [2002]. However, they are limited to money, process, and information flows. How each commitment operation influences the chain of commitments is not clearly specified. Such relationships are accommodated and exploited by our approach.

Economou et al.’s approach is similar to ours, but for a different purpose [2001]. Their premise is that the protocol is fixed. If deontic states are entered, then the commitments must be fulfilled. Since our approach captures the commitment causal relations and is intended to help for protocol design, the deontic states can always be changed. However, as the design is completed, the protocol is fixed, it becomes possible to accommodate some of the topics that Economou et al. study, e.g., inferring commitments from protocols.

Agent UML (AUML) provides various kinds of diagrams for a designer to specify multiagent system behavior and internal agent models [Odell et al., 2000]. AUML is weak in its handling of multiagent concepts and abstractions. Our commitment causality diagram can be a complementary, alternative way to specify system requirements. Adding CCDs to AUML would enhance the expressiveness of AUML to a commitment level.

It is helpful to consider three main classes of approaches for multiagent design for information management and electronic commerce applications.

- Flow-based. These approaches begin from a business requirement flow in which different tasks are processed in different stages and triggered by each other [O’Brien and Wiegand, 1998]. The agents are the task performers. Each agent takes inputs from its producers and sends outputs to its consumers. Although the agents cooperate to handle exceptions and ensure a reliable task execution, they lack autonomy and the ability to negotiate for a best solution. Any change of the business logic drastically affects each agent’s state machine and communication interface. Our approach emphasizes requirements as represented by commitments and causal relations among them. For example, whereas in flow-based design, a trip change made by a passenger may create several more tasks including canceling some reservations and updating the itinerary, in our approach, this change is treated as the update of a request. We can add edges among the affected commitments and trigger additional functions within the agents involved.
• **Service-based.** These approaches model each agent as a service provider [McIlraith et al., 2001]. Agents can receive and send service requests to each other by following the protocol specified for each service. Unlike the flow-based approaches, there is no predefined flow process, so agents never deliver services to others without being requested. However, a group of service providers can form a supply chain system if the services they provide are involved in a global business process. Service-based designs are a popular approach in today’s electronic commerce systems. However, in terms of the adaptability and flexibility, practitioners are still struggling for a better interaction model to make a supply chain more robust, because current approaches lack the abstractions to handle different protocols and data interfaces needed to accommodate different business situations. The key missing abstraction is commitments, which we highlight in our work.

• **Commitment-based.** These approaches not only specify the services that the agents provide, but also capture how the agents interact. Commitments can be treated as common states during agent interaction, which represent the agreements that are achieved in the interaction at any given point. Any update, fulfillment, cancellation, or modification of the commitments affects how the services are delivered. By capturing the commitments in supply chains, we can control the lifecycles of service deliveries and are able to handle most business processes. Our approach belongs to this category. The difference between our approach and previous agent approaches is that we capture causal relations among commitments that are essential to link series of related operations.

### 6. CONCLUSIONS

Given an agent interaction marked up with the various columns of a conversation table, as in Table 1, our approach first extracts commitments and commitment relations from it. It then generates the agent models by fine-tuning the relation diagrams. Our approach enables designers to model a commitment-centric agent system based on an understanding of how the agents interact with each other by making, fulfilling, and modifying commitments. The main idea of this approach is in designing the commitment lifecycle in terms of the causal relations among commitments. Each commitment must be taken care of by its creator and any operation performed on a commitment must be properly propagated to other commitments to ensure that inconsistencies do not arise.

Our approach fits into a methodology that involves a combination of heuristic reasoning by a human designer (e.g., creating and marking up example interactions; enriching a CCD by hand if desired; coding domain-specific rules) and algorithmic reasoning (e.g., deriving a CCD and generating agent models). In this way, it goes beyond pure methodological approaches, which concentrate on the heuristic element.

As it stands, this work is a paper approach. It would be important to incorporate it into existing tools so that its effectiveness can be evaluated on larger problems. Please note, however, that many practical cases are not necessarily richer in structure—they may have more roles and more messages, but the key distinctions are not much richer than in the example of this paper.

This work leads to some interesting directions for future work. One, we plan to complete interactions and agent models so as to help a designer detect and address potential exception conditions. Two, we can incorporate a discussion of the deeper aspects of commitments that are not accommodated in the above approach. These include a treatment of the context group as well as of three additional commitment operations (release, assign, and delegate) [Singh, 1999]. Doing so can enable us to capture richer multiagent systems and handle more realistic situations where they are applied. Three, we will formally evaluate the computational complexity of our construction algorithm.

### Acknowledgments

This work was supported by the National Science Foundation under grants IIS-9624425 and DST-0139037. We are indebted to Jie Xing for several useful discussions.

### References


