Mapping Dooley Graphs and Commitment Causality to the π-Calculus*

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

Commitments among agents can be used to model business processes in a manner that preserves the autonomy and heterogeneity of the interacting parties. Dooley graphs are widely used in conversation-based multiagent system design. Commitments and their causal relationships can be inferred from Dooley graphs to flexibly model business processes.

We present a theoretical foundation for Dooley graphs, commitments, and causality in terms of the π-calculus, a process algebra for specifying concurrent systems, which has found application in business process modeling. This paper expresses the key elements of Dooley graphs (roles, characters, conversations) and of our approach (commitments and causal diagrams) using the π-calculus. Further, we accommodate a new variety of primitives for exception handling termed reentrant connectors. The π-calculus enables us to derive useful properties from a given model and to validate its correctness. We show how these properties assist in building an entire business process model in which agents interact flexibly.

1. Introduction

Commitments are a key element in a multiagent system and especially in applications involving business processes [11]. Commitments record the obligations of an agent to another so that at any given stage we can tell what tasks have been done and what events are expected. When commitments are incorporated into business process specifications, interaction histories can be maintained and related processes can be brought together in a natural manner.

Commitments preserve agent autonomy and heterogeneity. Agents interact by manipulating commitments they make to each other. Thus each agent can actively make decisions based on its best interests but without exposing its internal behavior. Incorporating commitments into business process modeling helps the designer focus on high-level requirements and dependencies among partners instead of low-level process dependencies. The process enactment is guided by the semantics of commitment operations and conversations among agents. Therefore, a commitment-oriented business process model can more flexibly adapt to changes of business requirements and organizations than traditional approaches.

A Dooley graph visualizes an agent interaction as a set of conversations each of which addresses one particular purpose. Previous research has used Dooley graphs to derive agent behavior and skeletons. We previously showed how commitments and their casual relations can be derived from Dooley graphs [10]. This is based on the idea that conversations encapsulate the communication within a business process so that agents can flexibly execute different protocols while still maintaining commitments made to each other.

We formulate our commitment-based model using the π-calculus, a process algebra capable of specifying a concurrent system. The formalization provides not only a means of rigorously enacting the computations, but also provides the grounds for proving correctness as well as other interesting properties. Here, we mainly use the π-calculus to specify our model components such as reentrant connectors, commitment operations, and conversations. Most importantly by using the π-calculus, we will demonstrate how this formalization can help test, improve, and verify a system model.

This paper is organized as follows. Section 2 provides technical background on Dooley graphs, commitments, and the π-calculus. Section 3 expresses our model elements using π-calculus and also introduces reentrant connectors. Section 4 derives some useful theorems and properties. Section 5 concludes with a discussion.

* This research was supported by the NSF under grant DST-0139037.

AAMAS'04, July 19-23, 2004, New York, New York, USA.

2. Technical Background

Now we present some essential background on Dooley graphs, commitments, commitment-oriented design of multiagent systems, and the π-calculus.

2.1. Dooley Graphs

Agent interactions can be decomposed into a set of conversations in which only pairs of agents communicate. A conversation is created when an agent sends out a new request or question and ends when the request or the question is resolved. Parunak proposed visualizing agent conversations via Dooley graphs, which we illustrate below [5].

As a running example, consider a travel planning user scenario, based on [10], itself adapted from Parunak. A customer (or passenger P) calls his travel agent (T) to book a trip. Upon receiving the order, T sends requests to the airline (A), hotel (H), and car rental (R) agents to reserve air tickets, hotels, and cars, respectively. Eventually, the latter three agents will send back the results to T and T will confirm an itinerary to P. During these transactions, P may update his trip request (e.g., cancel the car rental) or H may cancel a hotel reservation. These exceptions result in more conversations to update the trip itinerary. Table 1 shows an example utterance sequence.

By executing the algorithm described in [7], we obtain a Dooley graph shown in Figure 1. In this Dooley graph, there are 8 conversations, $\{\chi_1$=$\{1,10\}$, χ_2=$\{2,4\}$, χ_3=$\{5,6,21\}$, χ_4=$\{3,8,14\}$, χ_5=$\{7,9,12\}$, χ_6=$\{11,13\}$, χ_7=$\{15,16,19,20\}$, χ_8=$\{17,18\}\}$, 16 characters (the vertices in the graph representing the distinct agent behaviors in different conversations), $\{\mathtt{P}_1$–$\mathtt{P}_3$, $\mathtt{T}_1$–$\mathtt{T}_8$, $\mathtt{A}_1$–$\mathtt{A}_2$, $\mathtt{H}_1$–$\mathtt{H}_2$, $\mathtt{R}\}$, 8 roles (abstraction of the characters denoted in the dotted circles), and 7 agents (concrete entities playing one or more roles), $\{P$, T, A_1, A_2, H_1, H_2, $R\}$. Note that agent T plays 4 roles and has 8 characters $\mathtt{T}_1$–$\mathtt{T}_8$, and A_1 and A_2 play the same role, which has the characters $\mathtt{A}_1$ and $\mathtt{A}_2$.

The interesting point about Dooley graphs to note from this example is that they separate out an agent into its roles and a role into its characters. The characters carry out highly commitment-specific conversations with the characters of other agents. This paper seeks to formalize characters, roles, agents, and conversations in the π-calculus.

2.2. Commitment and Causality

A commitment is an obligation from a debtor to a creditor about a particular condition. For debtor x, creditor y, and condition p, the commitment is notated $\mathsf{C}(x, y, p)$. If p is a simple proposition or an action, such as shipping goods or making payment, we call the commitment *unconditional*.

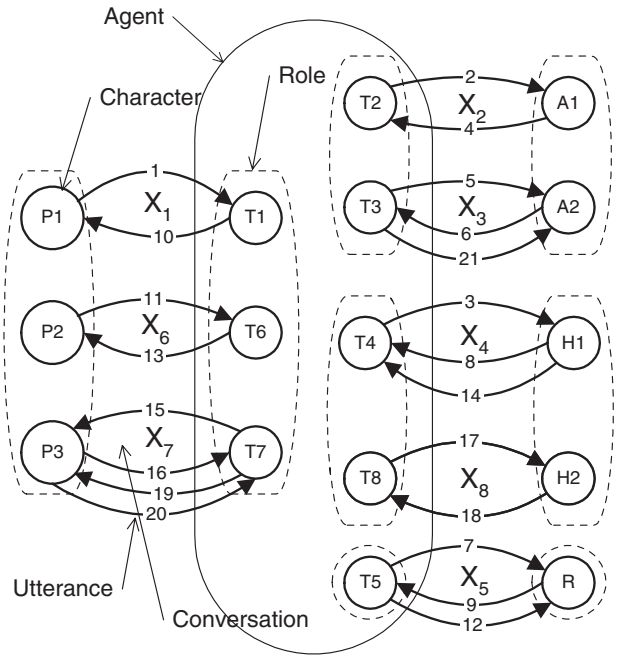


Figure 1. Dooley graph for trip planning

If p is in the form of $e \rightarrow q$ where q is a simple proposition (understood to be activated when e becomes true), then we call this type of commitment *conditional*. An example of a conditional commitment is when a customer promises to pay for a trip only if his travel agent books one for him.

Four operations drive the life cycle of commitments: Creation (Crt), Update (Upd), Discharge (Dcg), and Cancel (Cnl). A commitment is initially created when an agent makes a promise to another. The commitment is discharged when the promise is fulfilled, e.g., when the actions involved have been performed or the conditions have become true. Commitments can also be canceled but doing so would in general incur penalties. For example, if a customer cancels a hotel reservation less than 24 hours before check in, his credit card may be charged for the first night's stay.

Before commitments are discharged or canceled, they may also be updated. This often occurs when the debtor wants to change his input or commitment conditions to accommodate any changes that he encounters internally or externally. For example, a trip customer may want to change his flight or a travel agent may want to update an itinerary because of a canceled flight. Practically, all these changes only affect transaction details but do not break the original agreement made between the two agents. Updating commitments in such a manner enables us consolidate many processes under the same commitments and prevent business requirements from being artificially fragmented.

Whereas a single commitment represents an obligation

AAMAS'04, July 19-23, 2004, New York, New York, USA.

#	S	R	Act Type	Utterance	Respond to	Reply to	Resolve	Complete	Update
1	P	T	REQUEST	Book trip					
2	T	A_1	REQUEST	Buy ticket	1				
3	T	H_1	REQUEST	Reserve hotel	1				
4	A_1	T	REFUSE	Not Available	2	2	2		
5	T	A_2	REQUEST	Buy Ticket	4				
6	A_2	T	COMMIT	Confirm Ticket	5	5	5		
7	T	R	REQUEST	Rent car	1				
8	H_1	T	COMMIT	Confirm Hotel	3	3	3		
9	R	T	COMMIT	Confirm Car	7	7	7		
10	T	P	COMMIT	Send Itinerary	6, 8, 9	1	1		
11	P	T	REQUEST	Cancel Car from the Itinerary					1
12	T	R	CANCEL	Cancel Car	11			7	
13	T	P	COMMIT	Send Revised Itinerary	11	11	11		10
14	H_1	T	CANCEL	Cancel Hotel				8	
15	T	P	QUESTION	Alternate Hotel?	14				
16	P	T	INFORM	Yes	15	15	15		
17	T	H_2	REQUEST	Reserve Hotel	16				
18	H_2	T	COMMIT	Confirm Hotel	17	17	17		
19	T	P	COMMIT	Send Revised Itinerary	18	16	16		13
20	P	T	ACT	Pay for the trip	19	19		1	
21	T	A_2	ACT	Pay for the ticket	6, 20	6		5	

between a pair of agents, commitment causal relations reflect the agreements made among a group of agents and also reveal the chain of transactions in a business process. We derive commitments and commitment casual relations from Dooley graphs and use conversations to encapsulate the messages occurred between pairs of agents [10]. The outcomes of conversations are commitment operations. By capturing commitments and conversations, we partition a business process into a Commitment Causality Diagram (CCD), in which the business requirements are clearly specified in terms of commitments and their causal relations. A CCD enables multiagent system designers to focus on the consequences of agent conversations instead of low-level messaging details. It also enables runtime participants to manipulate these commitments by flexibly choosing conversations so that their autonomy and heterogeneity are preserved.

The following is a list of commitments derived from the running example. Figure 2 is the graphical representation of the system. It extends the CCD [10] by incorporating *conversation segments* and *connectors* (to be introduced below). For clarification, TBD means a condition is to be decided because of an incomplete sample conversation.

$$
\begin{aligned}
C_1 &= \mathsf{C}(P, T, \mathsf{C}(T, P, \textit{SendItinerary}) \rightarrow \textit{Pay}(P, T)) \\
C_2 &= \mathsf{C}(T, A, \mathsf{C}(A, T, \textit{Confirm}) \wedge \textit{Pay}(P, T) \\
&\qquad \rightarrow \textit{Pay}(T, A)) \\
C_3 &= \mathsf{C}(T, H, \textit{TBD}) \\
C_4 &= \mathsf{C}(T, R, \textit{TBD}) \\
C_5 &= \mathsf{C}(A, T, \textit{Confirm}) \\
C_6 &= \mathsf{C}(H, T, \textit{Confirm}) \\
C_7 &= \mathsf{C}(R, T, \textit{Confirm}) \\
C_8 &= \mathsf{C}(T, P, \textit{SendItinerary})
\end{aligned}
$$

2.3. π-Calculus Overview

The π-calculus is a process algebra capable of expressing mobile and concurrent systems [6]. The key concepts of the π-calculus are name binding and passing among concurrent processes. It provides four kinds of basic actions.

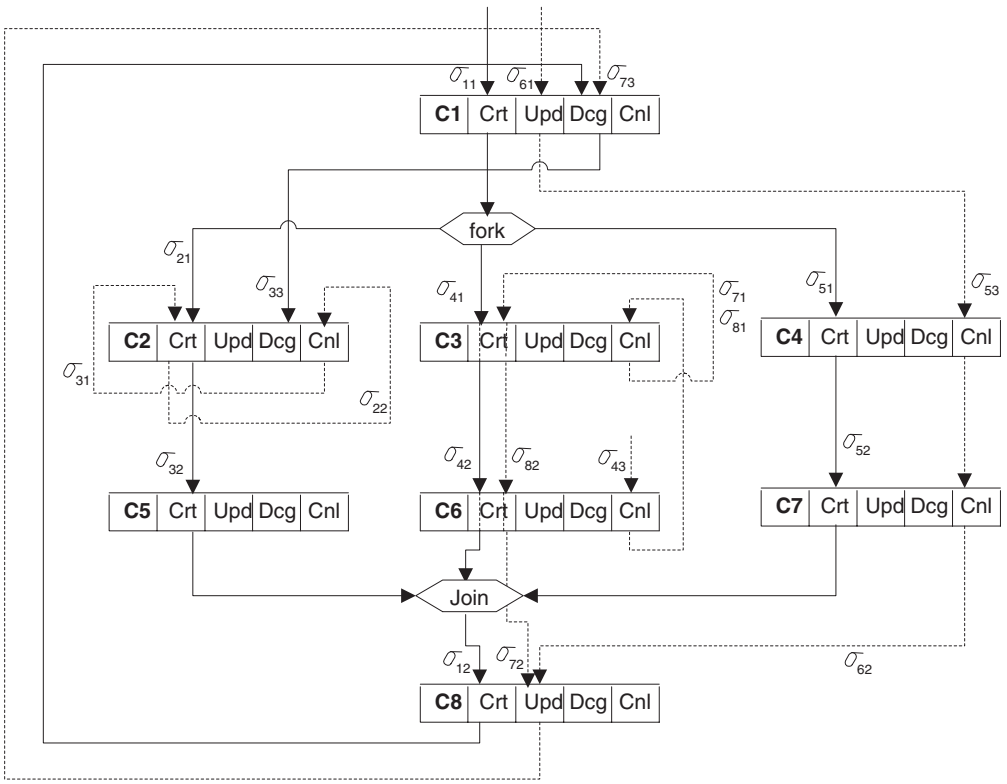
$$\pi \ ::= \ \overline{x}\langle\widetilde{y}\rangle \mid x(\widetilde{z}) \mid \tau \mid [\widetilde{z} = \widetilde{y}]\pi$$

Here $\overline{x}\langle\widetilde{y}\rangle$ sends tuple $\langle\widetilde{y}\rangle$ via name (or channel) x, $x(\widetilde{z})$ receives tuple $\widetilde{z}$ via name x, τ is an internal action not observable and $[\widetilde{z} = \widetilde{y}]\pi$ means perform capability π if $\widetilde{z}$ and $\widetilde{y}$ are same. These actions form the basis of a complete process syntax (defined in more detail next):

$$
\begin{aligned}
P &::= M \mid P \mid P' \mid \nu z P \mid !P \\
M &::= \mathbf{0} \mid \pi.P \mid M + M'
\end{aligned}
$$

- $\mathbf{0}$ is a null process which does nothing.
- $\pi.P$ is a process with π as the first action; process P cannot be executed before π is performed.
- $M + M'$ is a choice between M and M'. Only one of them will proceed and the other will be nullified.
- $P|P'$ denotes two concurrent processes P and P'. They can execute independently but can also interact with each other through shared names.

AAMAS'04, July 19-23, 2004, New York, New York, USA.



- $\nu z P$ restricts the use of name z in P so that it cannot be shared outside P.
- $!P$ is a repetition of P where $!P = P \mid !P$.

A common reduction of π-calculus, called PR-INTER, is shown below: $(\overline{x}\langle\widetilde{y}\rangle.P_1 + M_1) \mid (x(\widetilde{z}).P_2 + M_2) \rightarrow P_1|P_2\{\widetilde{y}/\widetilde{z}\}$. In this reduction, both action $\overline{x}\langle\widetilde{y}\rangle$ and $x(\widetilde{z})$ are performed and $\widetilde{z}$ is bound to $\widetilde{y}$.

For simplicity, we define the following notation to receive a constant $Const$ through channel a (as mentioned above, τ stands for an internal basic action):

$$a(Const) \stackrel{def}{=} a(y).[y = Const]\tau$$

The constants in our models are any utterance u_i and commitment C_j.

3. Formulating the Models

Now we show how each element of Dooley graphs can be represented in the π-calculus.

3.1. Characters

We use β to represent the process executed by characters and use the character name as its subscript. Since a conversation is a sequence of utterances, the character is a single process executing a sequence of actions. The syntax of β is $\beta ::= \pi.\beta \mid \mathbf{0}$, where π has the following three forms:

- $\overline{a}u$. Send utterance u through a channel named by the character's name a.
- $b(u)$. Receive utterance u through a channel named by the communicating character's name b.
- $\overline{Op_i}\mathsf{C}_i$. Send commitment C_i through channel Op_i where $Op_i \epsilon \{Crt_i, Upd_i, Dcg_i, Cnl_i\}$. The operation channels for each commitment are unique. This action notifies connectors (described in Section 3.6) that the commitment operation Op_i has performed on commitment C_i. $\overline{Op_i}\mathsf{C}_i$ is generated after a receiving action $b(u)$ in which utterance u maps to $\overline{Op_i}\mathsf{C}_i$. The mapping between utterances and commitment operations was studied previously in [10].

The character processes for the above example are

$$\begin{aligned}
\beta_{P_1} &= \overline{P_1}u_1.T_1(u_{10}).\overline{Crt_8}\mathsf{C}_8.\mathbf{0}\\
\beta_{P_2} &= \overline{P_2}u_{11}.T_6(u_{13}).\overline{Upd_8}\mathsf{C}_8.\mathbf{0}\\
\beta_{P_3} &= T_7(u_{15}).\overline{P_3}u_{16}.T_7(u_{19}).\overline{Upd_8}\mathsf{C}_8.\overline{P_3}u_{20}.\mathbf{0}\\
\beta_{T_1} &= P_1(u_1).\overline{Crt_1}\mathsf{C}_1.\overline{T_1}u_{10}.\mathbf{0}\\
\beta_{T_6} &= P_2(u_{11}).\overline{Upd_1}\mathsf{C}_1.\overline{T_6}u_{13}.\mathbf{0}\\
\beta_{T_7} &= \overline{T_7}u_{15}.P_3(u_{16}).\overline{T_7}u_{19}.P_3(u_{20}).\overline{Dcg_1}\mathsf{C}_1.\mathbf{0}\\
\beta_{T_2} &= \overline{T_2}u_2.A_1(u_4).\mathbf{0}\\
\beta_{T_3} &= \overline{T_3}u_5.A_2(u_6).\overline{Crt_5}\mathsf{C}_5.\overline{T_3}u_{21}.\mathbf{0}\\
\beta_{T_4} &= \overline{T_4}u_3.H_1(u_8).\overline{Crt_6}\mathsf{C}_6.H_1(u_{14}).\overline{Cnl_6}\mathsf{C}_6.\mathbf{0}\\
\beta_{T_8} &= \overline{T_8}u_{17}.H_2(u_{18}).\overline{Crt_6}\mathsf{C}_6.\mathbf{0}\\
\beta_{T_5} &= \overline{T_5}u_7.R(u_9).\overline{Crt_7}\mathsf{C}_7.\overline{T_5}u_{12}.\mathbf{0}
\end{aligned}$$

AAMAS'04, July 19-23, 2004, New York, New York, USA.

$$
\begin{aligned}
\beta_{A_1} &= T_2(u_2).\overline{Crt_2}\mathsf{C}_2.\overline{A_1}u_4.\mathbf{0}\\
\beta_{A_2} &= T_3(u_5).\overline{Crt_2}\mathsf{C}_2.\overline{A_2}u_6.T_3(u_{21}).\overline{Dcg_2}\mathsf{C}_2.\mathbf{0}\\
\beta_{H_1} &= T_4(u_3).\overline{Crt_3}\mathsf{C}_3.\overline{H_1}u_8.\overline{H_1}u_{14}.\mathbf{0}\\
\beta_{H_2} &= T_8(u_{17}).\overline{Crt_3}\mathsf{C}_3.\overline{H_2}u_{18}.\mathbf{0}\\
\beta_R &= T_5(u_7).\overline{Crt_4}\mathsf{C}_4.\overline{R}u_9.T_5(u_{12}).\overline{Cnl_4}\mathsf{C}_4.\mathbf{0}
\end{aligned}
$$

3.2. Roles

We use ρ to represent the process executed by a role. It is composed by the repetition of the choices of its character processes since at any time for a given role, only one character process can be engaged and the role can execute its character processes repeatedly. The subscripts of roles are the combination of two interacting characters. For example, one of the travel agent roles, $\langle T_1, T_6, T_7\rangle$, which interacts with role $\langle P_1, P_2, P_3\rangle$, is denoted as TP, and the corresponding customer role is denoted as PT. The role processes for the above example are

$$
\begin{aligned}
\rho_{\text{PT}} &= !(\beta_{P_1} + \beta_{P_2} + \beta_{P_3})\\
\rho_{\text{TP}} &= !(\beta_{T_1} + \beta_{T_6} + \beta_{T_7})\\
\rho_{\text{TA}} &= !(\beta_{T_2} + \beta_{T_3})\\
\rho_{\text{TH}} &= !(\beta_{T_4} + \beta_{T_8})\\
\rho_{\text{TR}} &= !\beta_{T_5}\\
\rho_{\text{AT}} &= !(\beta_{A_1} + \beta_{A_2})\\
\rho_{\text{HT}} &= !(\beta_{H_1} + \beta_{H_2})\\
\rho_{\text{RT}} &= !\beta_R
\end{aligned}
$$

3.3. Agents

We use α to represent the process executed by an agent that is composed by the actions performed by the roles it plays. The list of agent processes for the above example are

$$
\begin{aligned}
\alpha_P &= \rho_{\text{PT}}\\
\alpha_T &= \rho_{\text{TP}} \mid \rho_{\text{TA}} \mid \rho_{\text{TH}} \mid \rho_{\text{TR}}\\
\alpha_A &= \rho_{\text{AT}}\\
\alpha_H &= \rho_{\text{HT}}\\
\alpha_R &= \rho_R
\end{aligned}
$$

3.4. Conversations

A conversation is a sequence of utterances between two characters. We use ξ to represent the processes involved in a conversation. The conversation processes are the combination of the two interacting character processes. Consequently, the list of conversation processes for the above example are

$$
\begin{aligned}
\xi_{\chi_1} &= \beta_{P1} \mid \beta_{T1}, \qquad \xi_{\chi_2} = \beta_{T2} \mid \beta_{A1}\\
\xi_{\chi_3} &= \beta_{T3} \mid \beta_{A2}, \qquad \xi_{\chi_4} = \beta_{T4} \mid \beta_{H1}\\
\xi_{\chi_5} &= \beta_{T5} \mid \beta_R, \qquad \xi_{\chi_6} = \beta_{P2} \mid \beta_{T6}\\
\xi_{\chi_7} &= \beta_{P3} \mid \beta_{T7}, \qquad \xi_{\chi_8} = \beta_{T8} \mid \beta_{H2}
\end{aligned}
$$

3.5. Conversation Segments

A conversation segment is a partition of a conversation that has one of the following three forms, $P.\overline{Op}\mathsf{C} \mid P'$, $P \mid P'.\overline{Op}\mathsf{C}$ or $P \mid P'$, where process P and P' are sequences of actions that do not contain any commitment operations and all the utterances involved in P or P' only respond to utterances within P and P'. Conversation segments either result in a transactional activity (a commitment operation), such as reserving a hotel or confirming a ticket; or a nontransactional activity, such as a query for information or negotiation about price. The dependencies among conversation segments are decided by connectors, as described in the later sections.

We use σ to represent conversation segments and use both the conversation subscript and the segment number as the subscript of σ. We prefix each conversation segment with a receiving action TR_i to trigger its execution. The addition of the trigger action makes a conversation ξ differ from its segments. However, since the triggers are consumed when put in conjunction with connectors, we can consider conversations and segments as equivalent when reasoning with connectors. The conversation segments for our example are

$$
\begin{aligned}
\sigma_{11} &= \text{TR}_{11}.(\overline{P_1}u_1.\mathbf{0} \mid P_1(u_1).\overline{Crt_1}\mathsf{C}_1.\mathbf{0})\\
\sigma_{12} &= \text{TR}_{12}.(T_1(u_{10}).\overline{Crt_8}\mathsf{C}_8.\mathbf{0} \mid \overline{T_1}u_{10}.\mathbf{0})\\
\sigma_{21} &= \text{TR}_{21}.(\overline{T_2}u_2.\mathbf{0} \mid T_2(u_2).\overline{Crt_2}\mathsf{C}_2.\mathbf{0})\\
\sigma_{22} &= \text{TR}_{22}.(A_1(u_4).\overline{Cnl_2}\mathsf{C}_2.\mathbf{0} \mid \overline{A_1}u_4.\mathbf{0})\\
\sigma_{31} &= \text{TR}_{31}.(\overline{T_3}u_5.\mathbf{0} \mid T_3(u_5).\overline{Crt_2}\mathsf{C}_2.\mathbf{0})\\
\sigma_{32} &= \text{TR}_{32}.(A_2(u_6).\overline{Crt_5}\mathsf{C}_5.\mathbf{0} \mid \overline{A_2}u_6.\mathbf{0})\\
\sigma_{33} &= \text{TR}_{33}.(\overline{T_3}u_{21}.\mathbf{0} \mid T_3(u_{21}).\overline{Dcg_2}\mathsf{C}_2.\mathbf{0})\\
\sigma_{41} &= \text{TR}_{41}.(\overline{T_4}u_3.\mathbf{0} \mid T_4(u_3).\overline{Crt_3}\mathsf{C}_3.\mathbf{0})\\
\sigma_{42} &= \text{TR}_{42}.(H_1(u_8).\overline{Crt_6}\mathsf{C}_6.\mathbf{0} \mid \overline{H_1}u_8.\mathbf{0})\\
\sigma_{43} &= \text{TR}_{43}.(H_1(u_{14}).\overline{Cnl_6}\mathsf{C}_6.\mathbf{0} \mid \overline{H_1}u_{14}.\mathbf{0})\\
\sigma_{51} &= \text{TR}_{51}.(\overline{T_5}u_7.\mathbf{0} \mid T_5(u_7).\overline{Crt_4}\mathsf{C}_4.\mathbf{0})\\
\sigma_{52} &= \text{TR}_{52}.(R(u_9).\overline{Crt_7}\mathsf{C}_7.\mathbf{0} \mid \overline{R}u_9.\mathbf{0})\\
\sigma_{53} &= \text{TR}_{53}.(\overline{T_5}u_{12}.\mathbf{0} \mid T_5(u_{12}).\overline{Cnl_4}\mathsf{C}_4.\mathbf{0})\\
\sigma_{61} &= \text{TR}_{61}.(\overline{P_2}u_{11}.\mathbf{0} \mid P_2(u_{11}).\overline{Upd_1}\mathsf{C}_1.\mathbf{0})\\
\sigma_{62} &= \text{TR}_{62}.(T_6(u_{13}).\overline{Upd_8}\mathsf{C}_8.\mathbf{0} \mid \overline{T_6}u_{13}.\mathbf{0})\\
\sigma_{71} &= \text{TR}_{71}.(T_7(u_{15}).\overline{P_3}u_{16}.\mathbf{0}\\
&\qquad \mid \overline{T_7}u_{15}.P_3(u_{16}).\overline{\text{TR}_{81}}.\mathbf{0})\\
\sigma_{72} &= \text{TR}_{72}.(T_7(u_{19}).\overline{Upd_8}\mathsf{C}_8.\mathbf{0} \mid \overline{T_7}u_{19}.\mathbf{0})\\
\sigma_{73} &= \text{TR}_{73}.(\overline{P_3}u_{20}.\mathbf{0} \mid P_3(u_{20}).\overline{Dcg_1}\mathsf{C}_1.\mathbf{0})\\
\sigma_{81} &= \text{TR}_{81}.(\overline{T_8}u_{17}.\mathbf{0} \mid T_8(u_{17}).\overline{Crt_3}\mathsf{C}_3.\mathbf{0})\\
\sigma_{82} &= \text{TR}_{82}.(H_2(u_{18}).\overline{Crt_6}\mathsf{C}_6.\mathbf{0} \mid \overline{H_2}u_{18}.\mathbf{0})
\end{aligned}
$$

AAMAS'04, July 19-23, 2004, New York, New York, USA.

3.6. Reentrant Connectors

Traditional task connectors, e.g., as in traditional process modeling languages [2], specify the data or control flows among the business tasks that they connect. But in many business applications, the processes are long-lived and the components are autonomous service providers. These service providers can raise exceptions or generate updates based on their local reasoning. For example, a customer may update his trip itinerary or the hotel may cancel a reservation because of a problem at the intended facility. Existing models would treat such updates disjointly from the original execution, creating separate processes to handle them without explicitly relating them to the original transactions. Thus traditional approaches unnecessarily complicate the modeling and enactment of processes.

We introduce *reentrant connectors*, which not only control the initial process coordination requirement, but also deal with any updates sent or exceptions raised from the initial processes. The essence of our connectors is to allow processes to repeatedly deliver results through the same control logic until an entire transaction ends. To accomplish this, we use commitments as described above and let our connector to coordinate on commitment operations to ensure that commitment casual relations are not violated. The following are the differences between reentrant and traditional connectors.

- Both inputs and outputs of our connectors are the commitment operations which convey commitment casual relations instead of process dependencies.
- The decision-making process in our connectors consists not only of Boolean logic, but also triggers for conversations. By engaging in conversations, agents are able to negotiate and collaborate to find optimal business values based on their local decisions. Thus we preserve agent autonomy.
- The operations performed on a commitment are fed to the same connectors where the original commitment is involved. This reentrance under revisions enables better modeling and enactment of processes under exceptions and opportunities.

Figure 3 is the graphical illustration of a generic reentrant connector. Each connector contains two parts, one for initial creation and the other for revision. The initial creation process captures the business requirement level dependencies which maps to the service request and response chain. The revision process handles updates, cancellations, and terminations occurring along the above established chains. In our model, since we use commitment to capture the stages of agent interactions, the revision processes can be naturally attached to the existing commitments, thus making multiagent system modeling clear and extensible.

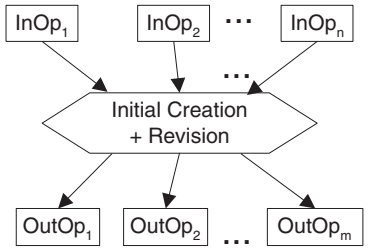


Figure 3. Reentrant connectors

Here we define three commonly used connectors, Γ_{join}, Γ_{fork} and Γ_l.

3.6.1. Join Connector Γ_{join} The join connector has multiple input commitment creation links and but only one output commitment creation link. Initially the join connector requires that all its input links have creation of the corresponding commitments; the output would be a commitment creation as well. After the initial join, commitment Update, Discharge, and Cancel may come in through any input link. For Update, the connector may output a commitment update or discharge. For Discharge, the connector will output a commitment discharge only when all the input links have Discharge operations present. For cancellation, depending on business rules, the connector may output commitment cancellation, update, or nothing.

As an example, the join connector executed by agent T on commitment C_5, C_6, and C_7 is expressed as follows,
$\Gamma_{join}(\langle \mathsf{C}_5, \mathsf{C}_6, \mathsf{C}_7\rangle \rightarrow \mathsf{C}_8) =$

$$\begin{array}{c}\nu d\,(\, Crt_5(C_5).\overline{S}d.\mathbf{0} \mid Crt_6(C_6).\overline{S}d.\mathbf{0} \mid Crt_7(C_7).\overline{S}d.\mathbf{0} \\ \mid S(x).S(x).S(x).\overline{\mathrm{TR}_{12}} \\ .!(\, Cnl_6(\mathsf{C}_6).\overline{Cnl_3}\mathsf{C}_3.\mathbf{0} + Crt_6(\mathsf{C}_6).\overline{\mathrm{TR}_{72}}.\mathbf{0} \\ + Cnl_7(\mathsf{C}_7).\overline{\mathrm{TR}_{62}}.\mathbf{0)} \\ )\end{array}$$

The above connector only reflects the control logic of our running example. Initially, the connector waits for all the three commitment creations (by consuming three $S(x)$) and then executes conversation segment σ_{12} which creates commitment C_8. After the initial creation, it enters an infinite loop in which it may receive the cancellation of C_6 or C_7, or another creation of C_6. It then executes conversation segment σ_{71}, σ_{72}, or σ_{62} as appropriate, which lead to either the creation of C_3 or C_6 or an update of C_8.

3.6.2. Fork Connector Γ_{fork} The fork connector has only one commitment creation input link but multiple output commitment creation links. Initially upon receiving a commitment creation on its input link, the fork connector triggers all commitment creations on its output links. After the initial execution, commitment Update, Discharge, and Cancel may come in through the input link.

AAMAS'04, July 19-23, 2004, New York, New York, USA.

For update, depending on business rules, it may affect part or all of its output commitments with either update or cancellation operations. For discharge, the connector should send commitment Discharges to all its output links. For cancellation, the connector should send cancellation to all its output links.

As an example, the fork connector executed by agent T on commitment C_1 is expressed as follows
$\Gamma_{fork}(\mathsf{C}_1 \rightarrow \langle \mathsf{C}_2, \mathsf{C}_3, \mathsf{C}_4 \rangle) =$

$$\nu d(Crt_1(C_1).\overline{S}d.\overline{S}d.\overline{S}d.!(Upd_1(C_1).\overline{\mathrm{TR}_{53}}) \\ \mid S(x).\overline{\mathrm{TR}_{21}}.\mathbf{0} \mid S(x).\overline{\mathrm{TR}_{41}}.\mathbf{0} \mid S(x).\overline{\mathrm{TR}_{51}}.\mathbf{0})$$

Upon receiving a creation of C_1, the fork connector triggers all three conversation segments σ_{21}, σ_{41}, and σ_{51}, which in turn create commitments C_2, C_3, and C_4, respectively. Later, there is an update of C_1 which is justified to trigger conversation σ_{53} only. The latter cancels commitment C_4.

3.6.3. Linear Connector Γ_l Linear connectors are the same as fork connectors but with only one output commitment creation link. As an example, the linear connector $\Gamma_l(\mathsf{C}_3 \rightarrow \mathsf{C}_6)$ is expressed as follows

$$\Gamma_l(\mathsf{C}_3 \rightarrow \mathsf{C}_6) \quad = \quad Crt_3(\mathsf{C}_3).\overline{\mathrm{TR}_{42}}.!(Cnl_3(\mathsf{C}_3).\overline{\mathrm{TR}_{71}}.\mathbf{0} \\ + Crt_3(\mathsf{C}_3).\overline{\mathrm{TR}_{82}}.\mathbf{0})$$

4. Reasoning on the Formulations

Using the formulations of all the elements given above, we now define the process for an entire system Υ, which is the combination of agent processes and connector processes: $\Upsilon ::= \alpha_1 | \ldots | \alpha_n | \Gamma_1 | \ldots | \Gamma_m$. From this definition, we can derive some useful properties and also verify system correctness.

4.1. Behavior Derivation

Given a commitment operation $Op(\mathsf{C})$ and a system Υ, we can derive consequent conversations and commitment operations. This gives a designer a means to verify whether the multiagent system works as designed. As an example, to derive the consequent commitment operations of $Cnl_6(\mathsf{C}_6)$ (the hotel cancels the room reservation), we can start the reduction process from the revision part of $\Gamma_{join}(\mathsf{C}_5, \mathsf{C}_6, \mathsf{C}_7)$ since this connector covers the control logic of C_6. To save space, we only show the processes that are used in each derivation step and use "..." to denote the remaining processes. We also put the action performed in each step on derivation arrows.
$Cnl_6(\mathsf{C}_6).\overline{Cnl_3}\mathsf{C}_3.\mathbf{0} \mid \ldots$

$\xrightarrow{\overline{Cnl_6}\mathsf{C}_6}$ $\overline{Cnl_3}\mathsf{C}_3.\mathbf{0} \mid Cnl_3(\mathsf{C}_3).\overline{\mathrm{TR}_{71}}.\mathbf{0} \mid \sigma_{71} \mid \ldots$

$\xrightarrow{\overline{Cnl_3}\mathsf{C}_3}$ $\overline{\mathrm{TR}_{71}}.\mathbf{0} \mid \mathrm{TR}_{71}.(T_7(u_{15}).\overline{P_3}u_{16} \mid \overline{T_7}u_{15}.P_3(u_{16}).\overline{\mathrm{TR}_{81}}) \mid \sigma_{81} \mid \ldots$

$\xrightarrow{\overline{\mathrm{TR}_{71}}}$ $\overline{\mathrm{TR}_{81}} \mid \mathrm{TR}_{81}.(\overline{T_8}u_{17} \mid T_8(u_{17}).\overline{Crt_3}\mathsf{C}_3) \mid \ldots$

$\xrightarrow{\overline{\mathrm{TR}_{81}}}$ $\overline{Crt_3}\mathsf{C}_3 \mid Crt_3(\mathsf{C}_3).\overline{\mathrm{TR}_{82}}.\mathbf{0} \mid \sigma_{82} \mid \ldots$

$\xrightarrow{\overline{Crt_3}\mathsf{C}_3}$ $\overline{\mathrm{TR}_{82}}.\mathbf{0} \mid \mathrm{TR}_{82}.(H_2(u_{18}).\overline{Crt_6}\mathsf{C}_6 \mid \overline{H_2}u_{18}) \mid \ldots$

$\xrightarrow{\overline{\mathrm{TR}_{82}}}$ $\overline{Crt_6}\mathsf{C}_6 \mid Crt_6(\mathsf{C}_6).\overline{\mathrm{TR}_{72}}.\mathbf{0} \mid \sigma_{72} \mid \ldots$

$\xrightarrow{\overline{Crt_6}\mathsf{C}_6}$ $\overline{\mathrm{TR}_{72}}.\mathbf{0} \mid \mathrm{TR}_{72}.(T_7(u_{19}).\overline{Upd_8}\mathsf{C}_8 \mid \overline{T_7}u_{19}) \mid \ldots$

$\xrightarrow{\overline{\mathrm{TR}_{72}}}$ $\overline{Upd_8}\mathsf{C}_8 \mid \ldots$

$\xrightarrow{\overline{Upd_8}\mathsf{C}_8}$ $\ldots$

Now we know the chained effects of $\overline{Cnl_6}\mathsf{C}_6$ is $\overline{Cnl_3}\mathsf{C}_3 \xrightarrow{\overline{\mathrm{TR}_{71}}\xrightarrow{\overline{\mathrm{TR}_{81}}}} \overline{Crt_3}\mathsf{C}_3 \xrightarrow{\overline{\mathrm{TR}_{82}}} \overline{Crt_6}\mathsf{C}_6 \xrightarrow{\overline{\mathrm{TR}_{72}}} \overline{Upd_8}\mathsf{C}_8$. Further, since σ_{71} and σ_{72} are the segments of conversation ξ_{χ_7}, and σ_{81} and σ_{82} are the segments of ξ_{χ_8}, and we have $\xi_{\chi_7} = \beta_{P3} \mid \beta_{T2}$ and $\xi_{\chi_8} = \beta_{T8} \mid \beta_{H2}$, we also know this transition involve four characters $\mathtt{P}_3$, $\mathtt{T}_2$, $\mathtt{T}_8$ and $\mathtt{H}_2$ and three agents P, T, and H.

4.2. Adding Business Logic

We can accommodate further business requirement variations by simply expanding the connector decision logics. For example, the customer may request a trip package (a bundled flight, hotel, and car) to get discount. If he cancels the car (reflected by the path $Upd_1(\mathsf{C}_1) \rightarrow Cnl_4(\mathsf{C}_4) \rightarrow Cnl_7(\mathsf{C}_7)$), the prices for his airline and hotel may be increased. We can modify the connector $\Gamma_{join}(\langle \mathsf{C}_5, \mathsf{C}_6, \mathsf{C}_7 \rangle \rightarrow \mathsf{C}_8)$ to implement this logic. In the consequent actions of $Cnl_7(\mathsf{C}_7)$, we introduce an additional action choice $\overline{Upd_2}\mathsf{C}_2.\overline{Upd_3}\mathsf{C}_3.\mathbf{0}$. Now the join connector becomes

$$\nu d\ (\ Crt_5(C_5).\overline{S}d.\mathbf{0} \mid Crt_6(C_6).\overline{S}d.\mathbf{0} \mid Crt_7(C_7).\overline{S}d.\mathbf{0} \\ \mid S(x).S(x).S(x).\overline{\mathrm{TR}_{12}} \\ .!(\ Cnl_6(\mathsf{C}_6).\overline{Cnl_6}\mathsf{C}_3.\mathbf{0} + Crt_6(\mathsf{C}_6).\overline{\mathrm{TR}_{72}}.\mathbf{0} \\ + Cnl_7(\mathsf{C}_7).(\overline{\mathrm{TR}_{62}}.\mathbf{0} + \overline{Upd_2}\mathsf{C}_2.\overline{Upd_3}\mathsf{C}_3.\mathbf{0})))$$

Note that the decision logic on which actions to take is not explicitly shown in the connector. In our model, we emphasize the potential commitment causality and conversation consequences, and leave local decisions to the agents. Doing so maximizes agent autonomy and heterogeneity.

4.3. Verifying Model Consistency

Definition 1 *Assuming that the life time of any commitment is finite, a system Υ is consistent if and only if any runtime commitment instance is eventually discharged.*

The definition says that for a successful execution of a business process, all the participants must eventually fulfill their commitments. This property is useful in a highly

AAMAS'04, July 19-23, 2004, New York, New York, USA.

distributed and heterogeneous environment where a system can only be validated by verifying the agents' commitments made to each other.

Theorem 1 *If all commitment creation paths form a tree and there is an edge from one of the commitment creations to the discharge operation of the root commitment, then the system Υ is consistent.*

Proof Sketch: *(Due to space limitations, we only give a description of the proof instead of reasoning using the π-calculus.) The root commitment will be eventually discharged since the commitment that discharges the root will be eventually created. Since all other commitments have a creation path coming from root, they too will be discharged eventually.*

This resembles a service request and response chain which is triggered by a single requester. When the root requester is satisfied by a commitment created where it is the creditor, it will discharge his commitment which in turn discharges the entire chain of commitments.

5. Discussion

Commitments have been widely recognized as key representation of agreements made among autonomous and heterogeneous agents in a multiagent system. Although much work has been done at conceptual level, applications of commitments to business processes have not been adequately studied. The main challenge is how to incorporate commitments and agent interactions into business process specification so that autonomy can be entertained by business participants. As one of the first attempts in this direction, our work modeled agent interaction by abstracting commitments and conversations and devised a semantics for reentrant connectors which facilitate business flow to be enacted not only correctly but also flexibly.

The π-calculus has found application in software engineering [1] and business process modeling [4]. These works use the π-calculus to model computations, but conceptually the computations they think of do not have any special features geared to open environments. In particular, they lack anything analogous to manipulable commitments or reentrance. We use π-calculus to provide a theoretical foundation to our commitment-based approach. This formalization better uses the power of the π-calculus to capture the essence of higher-level abstractions. Using the π-calculus helps us not only derive useful properties on the relations among characters, roles, agents, commitments, and connectors, but also verify the correctness of a model. Petri nets are another prominent modeling approach for concurrent systems [8]. We find the algebraic nature of the π-calculus to be more natural for representing commitments than the graph representation of Petri nets.

Business process and web service choreography standards have evolved for several years, e.g., see [3]. Existing standards tend to focus on method invocations and simple message exchanges. Exceptions cannot be perspicuously modeled with such abstractions. When we introduce commitments and reentrant connectors, we allow these processes to be correlated under same commitments and coordinated by the original connectors so as to reduce the complexities of the composition specifications. This makes the entire process coherent even as it is modeled in a manner that can be readily enacted by autonomous, heterogeneous agents.

Conclusions This paper describes a methodology to map Dooley graphs and commitment casual relations to π-calculus and demonstrates how this formalization help to derive useful properties and prove soundness of our commitment based models. This approach has opened a new direction on how to reason about multiagent systems and apply them to real world needs such as business process modeling. We have already begun elaborating these techniques to support the derivation of agreements among agents given certain sets of commitments among them [9]. In future work, we will integrate the commitment-based calculus into business process specification.

References

[1] F. Achermann. *Forms, Agents and Channels – Defining Composition Abstraction with Style*. PhD thesis, U. Berne, 2002.

[2] BPEL. Business process execution language for web services 1.1, May 2003.

[3] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the Web Services Web. *IEEE Internet Computing*, 6:86–93, 2002.

[4] B. Mehta, M. Levy, G. Meredith, T. Andrews, and B. Beckman. BizTalk server 2000 business process orchestration. *IEEE Data Engg. Bull.*, 24(1):35–39, Mar 2001.

[5] H. V. D. Parunak. Visualizing agent conversations: Using enhanced Dooley graphs for agent design and analysis. *Proc. 2nd Intl. Conf. Multiagent Systems*, pp. 275–282, 1996.

[6] D. Sangiorgi and D. Walker. *The π-Calculus: a Theory of Mobile Processes.* Cambridge U. Press, 2001.

[7] M. P. Singh. Synthesizing coordination requirements for heterogeneous autonomous agents. *Autonomous Agents & Multi-Agent Systems*, 3(2):107–132, Jun 2000.

[8] Wil van der Aalst and Kees van Hee. *Workflow Management Models, Methods, and Systems*, MIT Press, 2002

[9] F. Wan and M. P. Singh. Analyzing multiparty agreements with commitments. *Proc. AOIS@AAMAS*, 2004.

[10] F. Wan and M. P. Singh. Commitments and causality for multiagent design. *Proc. 2nd AAMAS*, 2003.

[11] J. Xing, F. Wan, S. K. Rustogi, and M. P. Singh. A commitment-based approach for business process interoperation. *IEICE Trans. Info. Syst.*, E84-D(10):1324–1332, 2001.

AAMAS'04, July 19-23, 2004, New York, New York, USA.