Agent Communication, Social Commitment Theory and Polycategories

Research assignment for CPSC 601.68, Fall 2008

Brett Giles
University of Calgary
2500 University Drive NW
Calgary, Alberta, Canada
brett.giles@ucalgary.ca

ABSTRACT

This paper consists of two parts. The first part discusses the formal specification of the CASA library. CASA is a java library implementing agent societies and communication under the umbrella of social commitments. The second part explores the possibility of describing social commitment based agent communication via polycategories.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification—*Formal Methods*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs—*Logics of programs*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical logic—*Lambda Calculus and related systems*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents*

General Terms

Theory

Keywords

Polycategory theory, Social commitments, Agent communication

1. INTRODUCTION

Formal specification assists implementors of a library to ensure the library is behaving as desired. Additionally, specifications give the library user an alternate documentation of goals and specifics of the library. Object-Z, (Smith 2000; Duke and Rose 2000; Smith 2008), based on the Z specification language (Spivey 1992), is widely used for formal specifications, including specifications for CORBA in (Kreuz 1998) and XSLT in (Yang et al. 2003). Additionally, Object-Z enjoys a formal semantics, (Smith 1995), and a formal mapping between Object-Z and UML (Booch et al. 1999; Kim and Carrington 2000).

The first part of this paper will start a continuation of the work of Flores and Kremer, as originally published in (Flores and Kremer 2001; Flores 2002), by expanding the Object-Z specification presented in those works. The goal of this continuation is to work towards a presentation of the details of the implementation of this specification as done in the java library CASA(Kremer 2008a).

Social commitment theory is an alternative to the FIPA semantics (see (Foundation for Intelligent Physical Agents 2002a)) for agent communication. While social commitments contain all the FIPA message types (e.g., request, inform, etc.), it differs intrinsically in the requirements for sending a message and the requirements for replying. In FIPA, there exist specific pre-conditions for an agent to be allowed to send a message, which depend heavily on the agent being implemented in a BDI model. For details of BDI, see (Foundation for Intelligent Physical Agents 2002b,a) or (Kremer 2008b). These preconditions lead to issues such as the agents needing to maintain an omniscient viewpoint of all agents and requiring an agent handle predicates with numerous repititions of "I believe you believe I believe ...".

Social commitment theory is independant of the actual implementation of the agent, which may be BDI based, state based or a combination of these. When using social commitments, messages lead to finite conversations which typically resolve in a short period of time. Agents may start conversations at any time, without having to consider external preconditions. Once a conversation is in progress, an agent is only bound to continue the conversation to an endpoint as defined by the social commitment rules of the agent's society.[1]

The second part of the paper will explore using polycategories to model agent communication.

Standard computing has enjoyed a categorical description since (Lambek 1969) showed a correspondence between types, propositions in logic and Cartesian closed categories (sometimes referred to as the Curry-Howard-Lambek correspondence). Recent work, in (Cockett and Pastro 2007; Pastro 2004), gives a polycategorical description of concurrent programming semantics. The description provides a two-tier logic — a basic one for message creation corresponding

[1]In this paper, we do not consider the added complications of misbehaving agents, which may violate commitments.

UCalgary course work 2008 Calgary, Alberta Canada

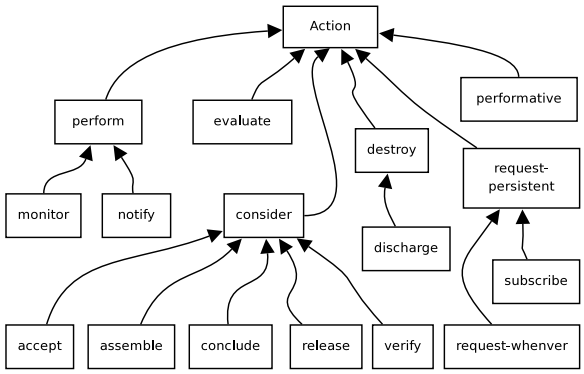

Figure 2: Acts for CASA

to classical computation and a second logic on top of that corresponding to message passing. As a prime component of agents is the ability to communicate via messages, this intrigued the author.

2. SOCIAL COMMITMENTS

Object-Z was used in (Flores and Kremer 2001; Flores 2002)to formally describe both the communications by agents and the social commitment rules used to create an operational society. In this section, the paper will first present an updated version of the specifications of the primary illocutionary points, then describe the current social commitment rules in CASA. It will conclude with a definition of an agent conversation, to be used below in section 3.

2.1 CASA message and action ontology

Messages in CASA(Kremer 2008a) contain both a *performative*, such as *inform, request* or *reply* and an *act* which is the action requested or referred to by the performative. Performatives are *illocutionary points* in speech-act theory.

In the formal definition of the CASA library, the social commitment rules depend on the message performative and the act. This may be considered the *type* of the message and as such is describable using the class feature in Object-Z. Since the original papers, Drs. Kremer and Flores have made a number of updates to the performatives modelled in CASA. The current performatives ontology in CASA is shown in diagram 2.1. A subset of the action ontology is shown in 2.

Note that all performatives are actions, but they are not included in 2 in this paper as the diagrams would be somewhat unwieldy . Additionally, because of the separation of the diagrams, there are some specific crossovers not shown directly on the figures. E.g., *Notify* is both a *Perform* and *Inform*, *Discharge* is both a *Destroy* and a *Request* and the performatives *RequestWhenever* and *Subscribe* are both *Request*s and *RequestPersistent*s.

2.2 Illocutionary Points

In (Flores and Kremer 2001), four main illocutionary points were defined: *Propose*, *Accept*, *Reject* and *Counter*. A fifth, *Inform*, is assumed in the paper.

In this section, we define the points *Inform*, *Petition*, *Request*, *Propose*, *Reply*, *NegativeReply*, *Agree*, *Refuse*, *Accept*, *Reject* and *Counter*. Note that with the exeption of *Counter* which is not defined in diagram 2.1, these names are drawn either directly from the current ontolgy of CASA or are obvious contractions of a name in the ontology.

Inform inherits from *ConversationalToken*, which is the basis of all illocutionary points (labelled as Performative in 2.1). *ConversationToken* inherits from *Data*, which is an abstract class that allows for any type of data to be carried in the conversation.

Inform

$$ConversationalToken$$

$$informing : \mathbb{P} \downarrow Data$$

Petition is the common basis of illocutionary points that ask another agent to do something, even just agreeing to the sending agent doing something. *Petition* consists of a reply-by time and an act that is being requested. Note that the class *Operation* applies to a set of *SocialCommitment*s and typically will add or delete commitments.

Petition

$$Inform$$

$$reply : Interval$$
$$act : \mathbb{P}_1 \downarrow Operation$$
$$act = informing$$

From *Petition*, there are two direct sub-classes, *Propose* and *Request*. The difference between the two is that *Propose* is used to offer a service to another agent and *Request* is used to ask an agent to perform a service.

Propose

$$Petition$$

$$proposing : \mathbb{P}_1 \downarrow Operation$$
$$proposing = act$$

Request

$$Petition$$

$$requesting : \mathbb{P}_1 \downarrow Operation$$
$$requesting = act$$

In response to messages, we can have either a *Reply*, which can be an *AffirmativeReply* or *NegativeReply*. *Reply* inherits from *Inform*.

Reply

$$Inform$$

$$replying : \mathbb{P}_1 \downarrow Operation$$
$$replying = informing$$

AffirmativeReply

$$Reply$$

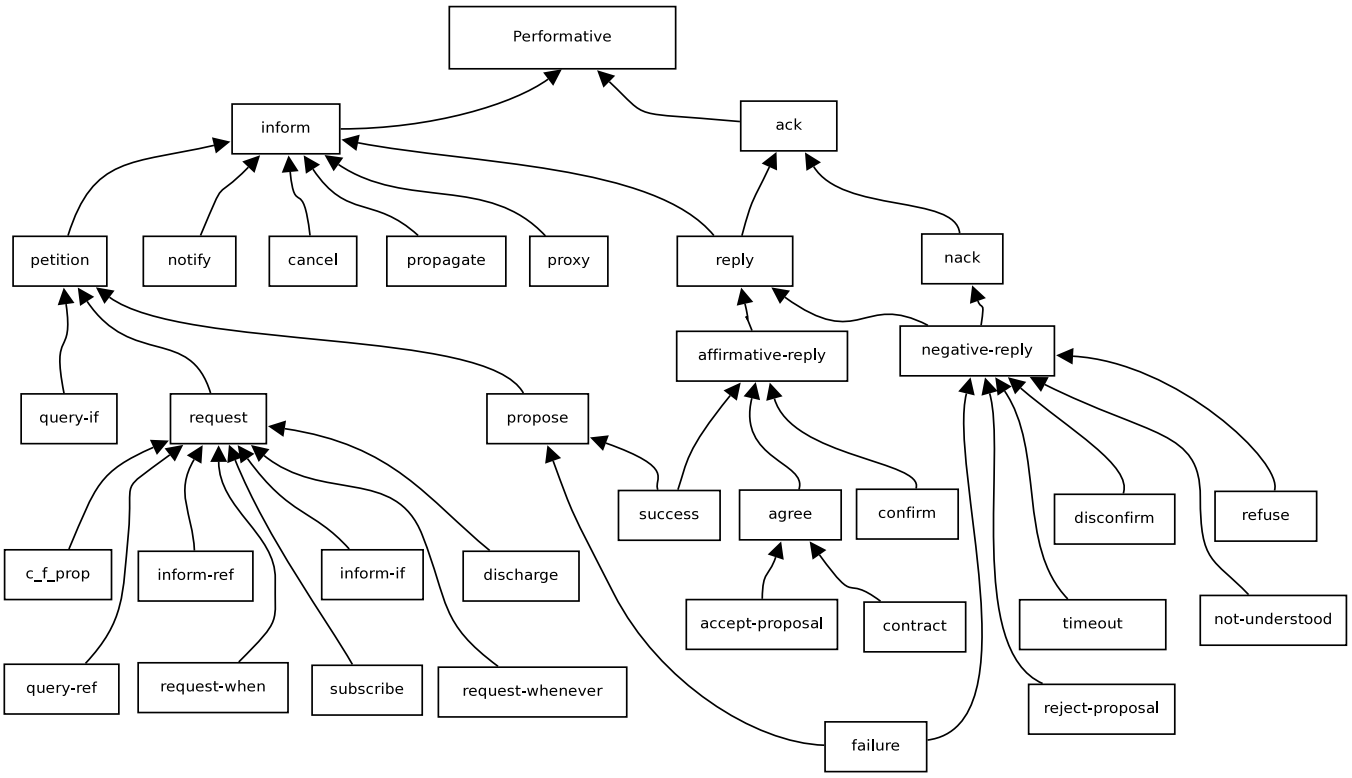

Figure 1: Message performatives for CASA

NegativeReply

$$Reply$$

The class *NegativeReply* has two "fail-safe" kinds of sub-types, *NotUnderstood* and *TimeOut*. These two may be used in reply to any kind of *Request* or *Propose* (and their sub-types).

NotUnderstood

NegativeReply

$$didNotUnderstand : \mathbb{P}_1 \downarrow Operation$$

$$didNotUnderstand = replying$$

TimeOut

NegativeReply

$$exceededTimeAllowed : \mathbb{P}_1 \downarrow Operation$$

$$exceededTimeAllowed = replying$$

The non fail-safe responses to a *Propose* may be one of *Accept* or *Reject*.

Accept

AffirmativeReply

$$accepting : \mathbb{P}_1 \downarrow Operation$$

$$accepting = replying$$

Reject

NegativeReply

$$rejecting : \mathbb{P}_1 \downarrow Operation$$

$$rejecting = replying$$

A non fail-safe response to a *Request* will be either an *AffirmativeReply* using *Agree* or a *NegativeReply* using *Refuse*.

Agree

AffirmativeReply

$$agreeing : \mathbb{P}_1 \downarrow Operation$$

$$agreeing = replying$$

Refuse

NegativeReply

$$refusing : \mathbb{P}_1 \downarrow Operation$$

$$refusing = replying$$

Finally, an agent may respond negatively to a *Request* or *Propose* and then give their own *Propose* via a *Counter*.

Counter
NegativeReply
Propose
proposing $\neq$ *replying*

2.3 Social commitment operators

In (Flores and Kremer 2001), the authors created a set of four policies to create commitments for proposals. This low number of policies was obviously a conscious choice made to allow the reader to see the essence of social commitment theory and not get bogged down in the details. These policies apply to messages in a hierarchy of illocutionary points.

In the current CASA implementation, there are a total of ten policies used when computing the commitments incurred or fulfilled by an agent while communicating. This subsection will present an expository description of these ten policies currently in use in the current implementation.

The original four classes that implemented the creation of social commitment operators were:

*PFP*1 - obligates the receiver of a proposal to reply with *Accept* or *Reject*.

*PFP*2 - fulfills the receiver's obligation to reply once they have replied.

*PFP*3 - add or remove commitments based on messages sent and received.

*PFP*4 - add a commitment to propose the discharge of an obligation.

2.3.1 Description of current CASA policies

In the current CASA library, some of the policies depend upon the *act* of the messages. See Figure 2 for a relevant subset of the current acts as defined in the ontology of CASA. By design, a specific message may have multiple policies that apply. This is effected by the hierarchy of messages. An example of this is given below in sub-section 2.3.2. An English description of each policy follows:

*CP*1**:** Applies to any message, fulfills any obligation the sender may have had to send this message.

*CP*2**:** Applies to an *Agree* to *Request* a *Discharge* of an action. Causes the fulfillment of the sender's commitment to *Request* (or *Petition*) to *Discharge* the action, the sender's commitment to perform the action and the sender's commitment to *Reply* to the original *Request* for the action.

*CP*3**:** Applies to an *Agree* in reply to a *Request* something. Adds a commitment on the sender to perform the something and a commitment to *Propose* the discharge of the perform of the something. cf. *PFP*4 and *PFP*3 above. [2]

*CP*4**:** Applies to an *Agree* in reply to a *Propose* a *Discharge* of an action. Also see *CP*2 above. Causes the fulfillment of the receivers commitment to *Propose* (also a *Petition*) to *Discharge* the action, the receiver's commitment to perform the action and the receiver's commitment to *Reply* to the original *Petition* for the action.

*CP*5**:** Applies to an *Agree* in reply to a *Propose* of an act. Adds a commitment for the receiver to perform the act and a commitment for the receiver to *Propose* a *Discharge* of the perform of the act. c.f. *PFP*4 above.

*CP*6**:** Applies to a *Cancel* of a *Subscribe* to any item. Causes the cancellation of the receiver's commitment to *Notify* the sender of the item cancelled and cancels the commitment of the receiver to *Monitor* the cancelled item.

*CP*7**:** Applies to a *Contract* of a *Subscribe* to anything. Adds a commitment to the sender to *Monitor* the item and a commitment to *Notify* the receiver about the item. This also adds a commitment on the receiver to *Cancel* the subscribe, which ensures that eventually the conversation will be terminated.

*CP*8**:** Applies to an *Inform*. This rule will create a commitment on the receiver to perform an *Action*. Typically, this is some form of *Reply* to the sender, but, for example, may be just a commitment to *Consider* a request. The type of act is dependent upon the type of the *Inform*. By default, the reply will be an *Accept*. See Table 1 for the specific type of *Reply* required depending on the type of the *Inform*.

*CP*9**:** Applies to a *Petition* of some act. This will add a commitment on the receiver to reply to the sender.[3]

*CP*10**:** Applies to a *Proxy* request. This will add a commitment on the receiver to proxy the request.[4]

2.3.2 Example application of policies to a message

Each of these policies has a specific type of message that it applies to. Note, however, that a specific message will likely have multiple policies apply to it. As an example, suppose Alice sends a *Request* to perform action *a* to Bob. From figure 2.1, we can see that a *Request* is a *Petition* which is a *Inform* which is a Performative (or *ConversationalToken* in the Object-Z description).

Because this is a message, *CP*1 will apply, fullfilling any commitment Alice had to actually send the request. Because the message is an *Inform*, of sub-type *Request* and the act is not *Discharge*, *CP*8 will apply and a commitment for Bob to *Consider* the request will be added. Finally, as this is a *Petition*, *CP*9 will apply and Bob will have a commitment to *Reply* to Alice, dependant upon the results of the *Consider*.

2.4 Agent conversation

Social commitment theory allows us to define an agent *conversation*. A *conversation* is the series of messages exchanged between two agents, starting with an initial message and terminating when all social commitments added through policies applied to messages in the conversation are fulfilled, cancelled or expired.

[2]CASA actually adds the proposal to discharge as a commitment that is dependent on the perform commitment.

[3]Note this is a dependent commitment, which depends on the receiver considering the act. This consider commitment will be generated by *CP*8.

[4]This is also dependent on the receiver's consider commitment, which will have been generated by *CP*8.

Inform sub-type	*MessageAct*	*Action* committed to
Cancel	-	*Release*
Notify	-	*Accept*
Propose	*Discharge*	*Release*
Propose	-	*Consider*
Proxy	-	*Assemble*
Reply	*Discharge*	*Conclude*
Reply	-	*Verify*
Request	*Discharge*	*Release*
Request	-	*Consider*
Petition	-	*Evaluate*
Inform	-	*Accept*

Table 1: *Inform* sub-types and *Reply* required

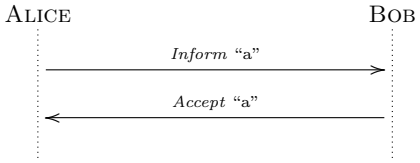

Figure 3: Inform Conversation

As examples, see figures 3 and 4. In figure 3, we see a two message conversation, starting with ALICE informing BOB of "a". The conversation terminates when BOB replies with an *Accept* message. In figure 4, the conversation consists of six messages, starting with the request from ALICE and terminating with BOB *Ack*nowledging that ALICE agreed that BOB has fulfilled the request.

3. AGENT COMMUNICATION AS A POLYCATEGORY

As noted in (Cockett and Pastro 2007), a polycategory provides a categorical model of the semantics of message passing. The intuitional basis for this is that processes (which are represented as the *polymaps* of the polycategory) may send and receive messages on multiple channels. These channels may be thought of as types and form the *objects* of the polycategory. Typical descriptions of this type of communication assume the messages (objects) are very low-level, for example, booleans or integers. More complex messages can be built up using type theory for polycategories as noted in (Cockett 2006).

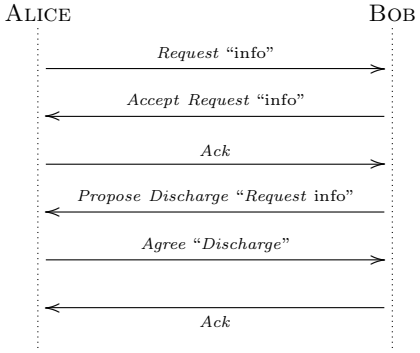

Figure 4: Request Conversation

3.1 Category theory preliminaries

This section will give a brief overview of the definitions and terminology made use of in this paper. For a more thorough introduction, please see (Barr and Wells 1995).

A *category*, $\mathbb{C}$, is defined as an entity having two collections of interest, its *objects*, $O_{\mathbb{C}}$, and its *arrows*, $A_{\mathbb{C}}$. These collections must obey the following rules:

- *Source, Target* There are total mappings $s, t : A_{\mathbb{C}} \to O_{\mathbb{C}}$. These map an arrow to its *source* and *target*.
- *Identities* For every $o \in O_{\mathbb{C}}$, there is a map 1_o whose source and target is o.
- *Composition* For every pair of arrows $f, g \in A_{\mathbb{C}}$ where the target of f is the source of g, there is another arrow, written as $f;\ g$ whose source is the source of f and target is the target of g.
- *Associativity* The composition of arrows is an associative operation.

Examples of categories include: SET where the objects are sets and the arrows are functions between them; GRP having groups as objects and the group homomorphisms between them as the arrows; A group may be considered a single object category with the elements of the group being the arrows of the category.

Categories are used to describe various mathematical objects, such as groups, rings, vector spaces, topological spaces and so forth. The typical algebraic structure of these items is described categorically in a variety of ways, including categorical descriptions of *products*, *sums*, *tensor products*, *subobjects* and others.

In the field of computing science, *computable functions* are of particular interest. In (Lambek 1969), it was shown that there is a three-way correspondence between these (as typed lambda calculus), intuitionistic logic and *Cartesian closed categories*. The latter type of category essentially means that products of objects and the functions between objects exist as other objects in the category. A more precise definition may be found in (Barr and Wells 1995) and others.

This correspondence provides a motivation for finding other categorical descriptions for alternate computing paradigms. Examples of this can be seen in the work (Cockett and Pastro 2007). The (Cockett and Pastro 2007) paper, and similar works, provide the motivation for our examination of agent communication and possible polycategorical interpretations.

3.2 Poly Categories

Ordinary categories provide a mathematical construction describing single maps between single objects. A *polycategory* provides a way to give a mathematical model of maps that connect multiple source and target objects at the same time.

From (Pastro 2004), a polycategory $\mathbb{P}$ is given by

- Objects $X_i, \ldots, Y_i \in \mathbb{P}_o$
- Polymaps which are given as:

$$\forall\, m, n \in \mathbb{N}, \quad \exists \text{ a set } \mathbb{P}(X_1, \ldots, X_m;\ Y_1, \ldots, Y_n)$$

which is the set of all polymaps from the Xs to the Ys.

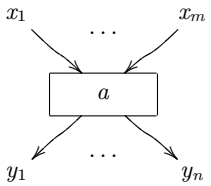

Figure 5: A polymap

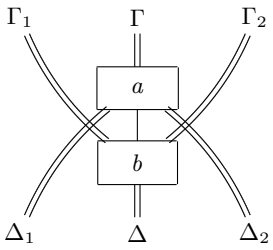

Figure 6: Polycategorical composition(cut)

- Identities: $1_X \in \mathbb{P}(X;\ X)$ for all $X \in \mathbb{P}_o$
- Composition (also referred to as *cut*) of polymaps is given by a map

$$C : \mathbb{P}(\Gamma;\ \Delta_1, y, \Delta_2) \times \mathbb{P}(\Gamma_1, y, \Gamma_2;\ \Delta) \rightarrow \mathbb{P}(\Gamma_1, \Gamma, \Gamma_2;\ \Delta_1, \Delta, \Delta_2)$$

  where Γ_1 or Δ_1 is empty and Γ_2 or Δ_2 is empty. This restriction is referred to as the *planarity condition*.

Note that the planarity condition on composition leads to 4 possible cases where composition may occur. In the above, the identities are true identity maps and the composition must satisfy associativity and interchange.

Interchange means that if a cut with x is followed by a cut using y, this is the same as first cutting with y and then with x. This occurs when you have a polymap in $\mathbb{P}(\Gamma_1;\ \Gamma_2, x, \Gamma_3, y, \Gamma_4)$ and it is being composed with a polymap from $\mathbb{P}(\Delta_1, x, \Delta_2;\ \Delta_3)$ and a polymap from $\mathbb{P}(\Phi_1, y, \Phi_2;\ \Phi_3)$. Regardless of which is done first, the result will be the same polymap in $\mathbb{P}(\Phi_1, \Delta_1, \Gamma_1, \Delta_2, \Phi_2;\ \Gamma_2, \Delta_3, \Gamma_3, \Phi_3, \Gamma_4)$.

A *symmetric polycategory* is one with a symmetry map $c_{\rho,\tau}$ where ρ and τ are permutations and $c_{\rho,\tau} : \mathbb{P}(\Gamma, \Delta) \rightarrow \mathbb{P}(\rho\Gamma, \tau\Delta)$. This is stating that the order of the objects coming into a polymap and coming out of a polymap is not important. The symmetry map is composable and must commute with cut.

3.3 Circuit diagrams

Circuit diagrams are an accepted way of visualizing and representing polycategories. See figure 5 and 6 for a representation of a polymap and the possible combinations for cut respectively.

Notice this is the opposite from naïve drawings explaining categories where the maps are typically drawn as arrows or connectors and the objects as boxes or circles. At the same time, note that this graphical representation seems, at the very least, relatable to agent communication.

Considering the circuit diagrams as a representation of agent communication, the first correspondence is that boxes would represent agents. The obvious initial choice for the connections would be messages. In the FIPA world, this is likely what we would have to stay with as we attempted to describe this in a polycategorical way. However, this paper's premise is that the proper level of abstraction for description via polycategories is *conversation*, as defined above in subsection 2.4. The reasons for this choice will be discussed below.

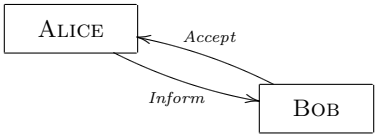

Figure 7: Polycategorical *Inform* with messages as objects

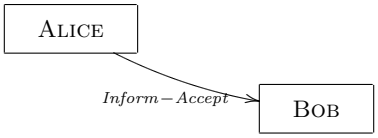

Figure 8: Polycategorical *Inform* with conversations as objects

3.4 Description of agent communication as a polycategory

The goal of this section is to provide a viable polycategorical description of agent communication. It will first discuss the issues associated with a naïve mapping consisting of messages as objects and agents as polymaps, followed by a presentation using conversations as objects and retaining agents as polymaps.

3.4.1 Problems with interpreting messages as objects

The initial appeal of this interpretation is that is seems to more closely match the standard polycategorical interpretation of message passing processes (see, e.g., (Cockett and Pastro 2007)). In fact, the basics of the interpretation (representation as circuits, Identities) are straightforward. For composition, use the time of the message to determine planarity and then cut is fully explicable.

The problem lies in that agents rarely (never when properly following social commitment theory as discussed in section 2) exchange a single message. As seen by the examples in figures 3 and 4, exchanges from start to finish consist of multiple messages. Consider the example of the *Inform* conversation where Alice sends the initial *Inform* to Bob. Bob then replies with an accept. This would appear like figure 7 in the polycategory and would not allow a *sensible* composition to be made.

Contrast this with the graphical representation of the same *Inform* conversation when the polycategorical objects are conversations, as shown in figure 8.

3.4.2 Interpretation with conversations as objects

This paper will refer to this potential polycategory as $\mathbb{AG}$.

First, $\mathbb{AG}_o$ will be the class of all conversations. Second, the polymaps will be agents. Conversations that are initiated by agents will be viewed as outputs of the agent polymap while conversations that are not initiated by an agent will be viewed as inputs to the agent polymap.

Empties	Cut is between ... (Refer to fig. 6)
$\Delta_2,\ \Gamma_1$	The last conversation started by a; which is the first conversation to b.
$\Gamma_2,\ \Delta_1$	The first conversation started by a; which is the last conversation to b.
$\Delta_1,\ \Delta_2$	The only conversation started by a; which is a conversation to b.
$\Gamma_1,\ \Gamma_2$	Some conversation started by a; which is the only conversation to b.

Table 2: Conversation cut / compositions

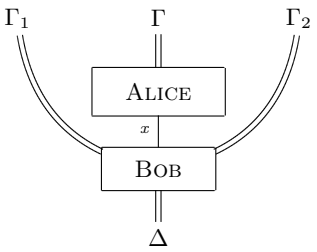

Figure 9: Alice composing with Bob

Identities.

Identities in polycategories are simply an identity map on a specific object. In agent communication, this would correspond to each conversation having a "pass-through" agent. Such agents would simply copy the messages of a conversation as it receives them.

Planarity condition and composition.

In order to define composition, the planarity condition for agent conversations must be defined. This paper contends that planarity for agent conversations is a timing constraint. Consider two conversations from Alice to Bob (ab) and Carol (ac) respectively. A conversation occurs during an interval of time — from the first message sent by Alice to the last message of the conversation. If the time interval of ab overlaps the time interval of ac, the object (conversation) connecting Alice with Bob will cross the object (conversation) connecting Alice with Carol and the resulting graph will *not* be planar.

For the four possible composition cases in a polycategory, this gives us table 2.

As an example, consider the agents Alice and Bob, where Alice is the receiver in the set of conversations Γ and starts only *one* conversation x, which is with Bob. Bob receives the set Γ_1 of conversations from other agents before participating in x and then receives the set of conversations Γ_2 after participating in x. During his lifetime, Bob also starts the set of conversations Δ with other agents. Composition then tells us there is an agent, AliBob, which is the recipient of the set of conversations $\Gamma_1 \cup \Gamma \cup \Gamma_2$ and starts the set Δ of outgoing conversations. Obviously, this agent can be physically implemented as a code combination of the two separate agents. This situation is shown in figure 9.

Given the physical implementation of composition (combining the agents), associativity and exchange follow immediately.

Symmetry.

Recalling that symmetry essentially states that we can permute the input and output conversations of a polymap, the immediate conclusion is that $\mathbb{AG}$ would not be a symmetric polycategory. To see this, consider the agent Bob who *Accept*s an *Inform* message regarding some external state t. Further suppose that if Bob has received this *Inform* message, he will respond to a *Request* about t with an *Accept*, but if he has not received the *Inform*, he will respond with a *Refuse*. Hence, the agent Bob (the polymap) will exhibit different behaviour depending on whether he receives the *Inform* before or after the *Request*.

As the above argument is obviously not mathematically rigorous, this is not a proof that agent conversation is not symmetric. Our intuition is that symmetry would not make sense in this setting.

3.5 Conclusions regarding the polycategorical interpretation

In practice, the conversations that agents participate in are not fixed. One might argue that a prime requirement of being an agent is that it be able to interact in a variety of conversations, occurring in varying orders and at varying times. This would seem to imply that a polycategorical description of an agent actually applies only to the particular instance of execution with a fixed set of input and output conversations. However, this seems to be not significantly different from the situation of representing programs and types via closed cartesian categories, (Lambek 1969).

A further question, however, is: What understanding does this give us of agent communication and could it be useful in the design of such systems? At this point, the author is unsure whether further investigation would be a fruitful line of research or not.

4. CONCLUSIONS

4.1 What has been accomplished?

In (Giles 2008), the author proposed three things:

1. Update the Object-Z description of the CASA library.
2. Enhance the specification of the library in some chosen area.
3. Explore whether there might be a polycategorical description of agent communication.

Work on proposal point 2 was not attempted, primarily due to time constraints. Work on proposal point 1 led to the exposition in section 2 where the author further described the current implementation of the social commitment policies in use in CASA, partially using Object-Z and partially via descriptive text and examples. The author views this as interesting work in its own right and a necessary precursor to fully satisfying proposal point 1. The work in section 3 has made significant headway on proposal point 3. While the work is somewhat expository in nature, I feel the decision to interpret conversations, rather than messages, as the objects of the polycategory as a key insight.

4.2 Further potential work

The first obvious piece is to further satisfy proposal point 1, that is, to revise and update the specifications of the social commitments protocol. The addition of dependent

social commitments appears to be an area that would be especially interesting to describe via Object-Z.

Second, the polycategorical representation of agent communication could be explored more rigorously. It would be interesting to continue the work of section 3 further, subjecting it to more mathematical rigor and understanding how the semantics of message passing would relate to this interpretation, if at all. Another interesting avenue for exploration would be how to satisfy the planarity condition in a way that retains a meaningful composition and makes the interpretation a symmetric polycategory.

5. ACKNOWLEDGMENTS

The author gratefully acknowledges the use of the wizard type checker (Johnston 1996a,b), which was used to verify all Object-Z appearing in this paper.

The author also gratefully acknowledges informal discussions regarding message structure and polycategories with Robin Cockett(Cockett 2008).

References

Michael Barr and Charles Wells. *Category Theory for Computing Science*. Number 0. Prentice Hall, 2nd revised edition edition, 1995. ISBN 0133238091.

Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language Reference Manual*. Addison Wesley, 1999.

J. Robin B. Cockett. Informal conversations regarding message structure. Whiteboard talks, December 2008.

J. Robin B. Cockett. What is a good process semantics?, 2006. Available online at http://pages.cpsc.ucalgary.ca/~robin/talks/estonia.pdf.

J. Robin B. Cockett and Craig A. Pastro. The logic of message passing. *CoRR*, abs/math/0703713, 2007.

Roger Duke and Gordon Rose. *Formal Object-Oriented Specification using object-z*. Cornerstones of Computing. MacMillan Press Limited, 2000.

R. A. Flores and R.C. Kremer. To Commit or not to Commit: Modelling Agent Conversations for Action. Computational Intelligence. *Special Issue on Agent Communication Languages*, 18(2), 2001.

Roberto A. Flores. *A Theory of Software Agent Conversations*. PhD thesis, Department of Computer Science, University of Calgary, Canada, 2002.

Foundation for Intelligent Physical Agents. FIPA Communicative Act Library Specification. Technical report, FIPA, 2 December 2002a. Available: http://www.fipa.org/specs/fipa00037/SC00037J.pdf.

Foundation for Intelligent Physical Agents. FIPA ACL message structure specification. Technical report, FIPA, 2 December 2002b. http://www.fipa.org/specs/fipa00061/SC00061G.pdf.

Brett G. Giles. Research proposal for agent communication. Class proposal, October 2008.

Wendy Johnston. A type checker for object-z. Technical Report 96-24, The University of Queensland, 1996a.

Wendy Johnston. Wizard type checker. ftp://ftp.cs.uq.edu.au/pub/SVRC/software/wizard.tar.gz, 1996b.

Soon-Kyeong Kim and David Carrington. *A Formal Mapping between UML Models and Object-Z Specifications*, volume 1878 of *Lecture Notes in Computer Science*, pages 2–21. Springer Berlin / Heidelberg, 2000.

Rob Kremer. Casa - javadocs, October 2008a. `http://pages.cpsc.ucalgary.ca/~kremer/casa/index.html`.

Rob Kremer. Cpsc 601.68: Agent communication, September 2008b. `http://pages.cpsc.ucalgary.ca/~kremer/courses/AC/F2008/index.html`.

Detlef Kreuz. Formal specification of corba services using object-z. *Formal Engineering Methods, International Conference on*, 0:180, 1998.

Joachim Lambek. *Category theory, homology theory and their applications*, volume 86 of *Lecture Notes in Computer Science*, chapter Deductive systems and categories: II: Standard constructions and closed categories, pages 76–122. Springer Berlin / Heidelberg, 1969.

Craig A. Pastro. ΣΠ-polycategories, additive linear logic and process semantics. Master's thesis, Department of Computer Science, University of Calgary, Canada, 2004.

Graeme Smith. Object-Z home page, 2008. Available online at http://www.itee.uq.edu.au/~smith/objectz.html.

Graeme Smith. *The Object-Z Specification Language*. Kluwer Academic Publishers, 2000. ISBN 0-7923-8684-1.

Graeme Smith. A fully abstract semantics of classes for object-z. *Formal Aspects of Computing*, 0:30–65, 1995.

J.M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, 1992. Available online at http://spivey.oriel.ox.ac.uk/~mike/zrm/.

Hong Li Yang, Jin Song Dong, Ke Gang Hao, and Jun Gang Han. *Formalizing Semantics of XSLT using Object-Z*, page 595. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003.