TR-0829

協調システムの通信のためのメッセージとプロトコル [Messages and Protocols for Cooperative Systems Communication]

by

Stephen T.C. Wong

January, 1993

協調システムの通信のためのメッセージとプロトコル

Messages and Protocols for Cooperative Systems Communication

王 天賜 *

Summary. A cooperative problem solving (CPS) system refers to several loosely connected and potentially heterogeneous agents that cooperate to solve problems that require their combined expertise and resources. The purpose of this paper is to present key features of a communication scheme COSMO that has been used to support **cooperative problem solving** within a network of knowledge-based systems. We first propose two key design principles of this scheme: (1) the **loose coupling** of communication issues and knowledge representation issues, and (2) the notion of **communicative acts.** We then work these ideas into a set of basic components of COSMO which includes knowledge handlers, an operation model, organizational roles, message types, and communication protocols.

1 Introduction

The purpose of this paper is to present the key messages and negotiation protocols of COSMO – a communication scheme that supports orderly and rational interaction among cooperating agents. This scheme originated from the development of the Building Design Network (BDN) in the ATLSS Engineering Research Center at Lehigh University [ATLSS 92]. BDN is a distributed knowledge-based system prototype in which several agents with different construction expertise cooperate to obtain the preliminary building design. BDN agents reside in separate UNIX workstations and use TCP as the communication backbone. The knowledge bases are written predominantly in an object-oriented Prolog language [Wong 92b]. A more detailed description of COSMO is presented in [Wong 92].

The first design principle of COSMO is to use a **global language** of communication among heterogeneous agents. The use of a global language makes

* Stephen T. C. Wong. ICOT

fewer assumptions (thus it enables looser coupling) about how knowledge is represented in individual agents. Another advantage is that it enables the integration of preexisting, autonomous knowledge-based systems for broader applications. This also protects an organization's investment in local knowledge base management software, application programs, and user training.

To support a global communication language, an agent should equip with a **knowledge handler** for coding and decoding messages, in addition to a reasoning program (i.e., knowledge base and inference engine). In COSMO, such a knowledge handler is an independent computation process and normally runs as a background process. Such a handler also performs other crucial transaction functions, such as checking against possible errors and keeping track of pending messages. However, the use of knowledge handlers to facilitate the interface among heterogeneous agents is still not flexible enough for the interchange of knowledge. Cooperating agents must be able to communicate in a more verbalized and regulated way, and to convene various intentions during cooperation. In this respect, **speech act theory** [Searle 85] offers some insight.

Speech act theory states that the primitive units of human communication are speech acts of a certain type called **illocutionary acts**. Some examples of these are statements, questions, commands, promises, and apologies. Whenever a speaker utters a sentence in an appropriate context with certain intentions, he or she performs one or more illocutionary acts. In general, an illocutionary act consists of an illocutionary force F and a propositional content P. A special class of sentences that express elementary illocutionary acts of form F(P) are the performative sentences. These sentences consist of a performative verb used in the first person present tense of the indicative mood with an appropriate complement clause, i.e., a propositional content. In uttering a performative sentence one performs the illocutionary act with the illocutionary force named by the performative verb by way of representing oneself as performing that act. Some examples (with the performative verbs italicized) are: "I *promise* that I will do it tomorrow", "I *order* you to report the schedule to the project manager."

For distributed systems computation, a basic unit of communication among agents is the transfer of a message from one agent (the sender) to another (the receiver). The purpose of communication is to provide the receiver with some information or to have the receiver take certain actions. Inspired by the speech acts theory, such a unit is called a communicative act.

The second design principle is that a communication act is analogous to an **elementary** performative act in human communication; its message type τ expresses an illocutionary force F, named by a performative verb; and its message content p expresses a complement clause P, which is a specification of the sender intended to be computed by the receiver. Such a specification, as stated earlier, is represented in a format understandable by all agents.

Meanwhile, a message type τ has two basic functions. In the first place, it is an index for a receiver to select a procedure to compute the message content and to determine the type of the response act. Thus, the meaning of the performative verb that a type denotes is defined by the operation of the procedure associated with it. To maintain the consistency of its performative meaning, each of the message types defined in a cooperative system must indicate the same procedure across all agents. Secondly, a message type has a name similar to the performative verb that it denotes. For instance, a communication act of type *order* would tell a receiver to do something without the option of refusal. This act assumes that the sender has a higher authority or priority than the receiver. We believe that the use of such a logical naming convention to represent the concepts and structures inherent in a communication scheme impacts on the very thinking that goes into constructing that scheme.

In the subsequent sections, we present the basic components and some high-level protocols of COSMO that build on these two principles. Section 2 discusses four key components of the scheme. Section 3 describes some of the communication protocols devised to resolve conflict of knowledge among agents. The final section presents the conclusion.

2 Basic COSMO Components

2.1 Operational model

In COSMO, we consider two major kinds of communicative acts: those used to initiate actions and those used to respond to these actions. For an agent to know when to initiate an act A and whether that act is successful, we use the following **operational model** of communicative acts, {precondition}: A: {postcondition}, where {precondition} denotes a set of internal constraints which must be satisfied before A can be performed and {postcondition} denotes a set of conditions which must be met in order to consider that A is successful. Such a model is similar to the representational formalism used in most plan recognition algorithms. The representation formalism specifies that a precondition needs to be true to carry out the planning operation (a communicative act in our case) and an effect that holds once the operation is accomplished. The model of COSMO extends this formalism into distributed systems communication and enriches it with the concepts of the degree of strength and the classes of communicative acts.

For {precondition}, usually, it is the reasoning program of the agent which sometimes a reasoning program can have several alternative actions, and deciding which one to perform becomes a problem. For example, suppose that an agent wants to know something about p, but does not know which agent it should ask and is reluctant to broadcast the request in order to avoid excess communication. Its knowledge handler would then assign every feasible alternative a number and select the alternative with the highest number. Such a number is called a **utility** value. A set of heuristics for setting utility values in the knowledge handler for various implicit communication purposes is described in [Wong 92].

To terminate an act A properly, its {postcondition} requires that the sender must receive a message that either is a direct response of defined types to A or is a control message indicating certain communication problems.

2.2 Organizational roles

Organization hierarchies are used in cooperative problem solving systems to: (1) establish the problems to be solved; (2) segment the problems into separate activities to be performed by different agents; and (3) coordinate activities and tasks among agents so that overall solutions are achieved. Depending on the application, (1) and (2) may be pre-determined or may be jointly decided by the agents in the course of communication. Generally, there is a set of admissible roles in such a cooperative system, and each agent in the system is assigned one of these roles. The function of the roles is to indicate the position of that agent in the hierarchy and to determine what reasoning strategies to use. To compare the ranking differences of agents, this proposed scheme assigns a number to every role. For example, for two agents, a of role_a and b of role_b, agent a ranks higher than b if and only if (iff) $\mathrm{v}(\mathrm{role}_a) > \mathrm{v}(\mathrm{role}_b)$, where $\mathrm{v}(\mathrm{role}_x)$ denotes the role value of an agent x. In COSMO, the information of the organizational structure is encoded in a **role table**, which contains information about all agents' roles and their role values in the organization at a particular stage of operation.

The ranking difference between any two people in an organization affects their interactive behavior, such as decision making and communication. Any cooperative computing system that claims to exhibit certain human problem-solving abilities should exhibit such adaptive behavior. One way to accomplish this, as is implemented in our prototypes, is to have every agent partition its set of problem-solving strategies into several classes. An agent selects a particular class of strategies based on the ranking difference between itself and the would-be receiver. As an example, suppose that the set of strategies of agent a is $\mathrm{S}_a = \{\mathrm{s}_1, \cdots, \mathrm{s}_n\}$, where the subscripts are the indices of individual strategies. Then, agent a chooses s_i in S_a to interpret an incoming message from agent b when $v(role_a) - v(role_b) = i$. In this way, one can say that an agent has several classes of problem-solving strategies in its reasoning program and switches among them according to which agent it communicates with. The subsection below provides an example on the use of organizational roles to decide the types of communicative acts.

2.3 Message types

In COSMO, agents use two disjoint sets of message types: one set contains types that are strictly used in communicative acts for initiating actions, and the other contains types that are strictly used in response to the former acts. Hence, these response types are a postcondition of the acts of the first types. We call the messages in the first set, **action messages**, and those in the second set, **response messages**.

COSMO classifies message types according to the performative intents or purposes of their associated acts. It further distinguishes the types of a class to indicate the role relationship between the sender and the receiver. The BDN application, for example, has three classes of message types: inquiring, informing, and complaining (see Figure 1). Inquiry messages have two uses. To inquire is either to query for information or to request some action. Informative messages are both assertive and directive. To inform is either to give out information or to instruct someone to do something. The content of an informative message must be in grounded form, that is, without variables, while there is no such restriction for inquiry messages. Complaint messages are used to express one's dissatisfaction. To express dissatisfaction with a state of affairs commits the sender to presuppose both the existence of that state of affairs and that this particular state is bad for the sender.

Different communicative acts can sometimes achieve the same performative function with greater or lesser degrees of strength, e.g., suggesting that the receiver abort the task is weaker than ordering it to abort the task [Searle 85]. Following the discussion in Section 2.2, we have a role relation between the two communicating agents that dictates this degree of strength in COSMO. Here, we, in turn, use appropriate message types to indicate role relations explicitly. Figure 1 illustrates this point with a set of communicative acts that are currently implemented for the BDN application.

We briefly comment on the general usage of these message types here. Let degree(τ) represent the characteristic degree of strength of an act of action type τ. For inquiring acts, as an example, we have degree(*direct*) > degree(*ask*) > degree(*request*), as indicated by the role relations of the sender and the receiver. Such a comparison of message types has many implications

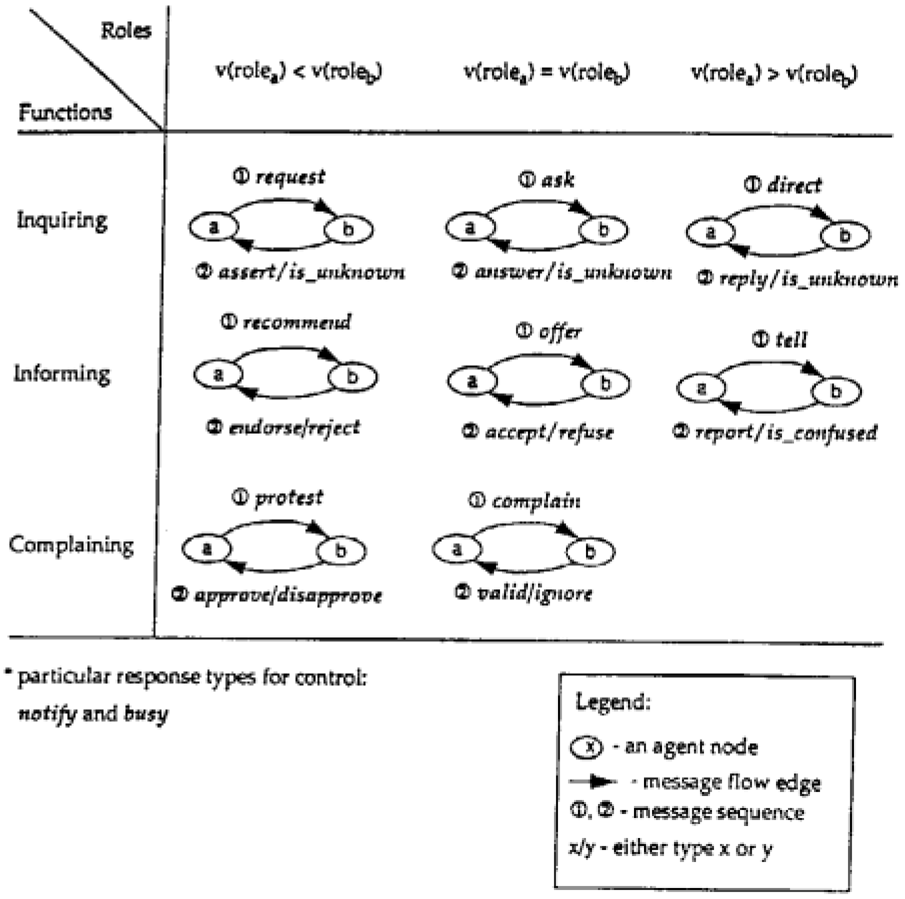

Figure 1. Communicative acts of COSMO implemented in BDN.

in cooperative systems communication. For instance, an act of *request* or *ask* type lets the receiver know that the sender is either of the same rank or a lower rank. Thus, the receiver can grant or refuse the inquiry by returning messages with either an assert or answer type. In a *direct* act, however, such a refusal is precluded as it is coming from a higher authority (see Figure 1). Or, to resolve the ordering conflict of messages, when *direct*(p) and *ask*(p') arrive at the same time from two remote agents, the local agent would execute the former message first. If an inquiry could not be understood or computable, the receiver would then simply send back an *is_unknown* message. Similarly, for informative acts, an agent can accept or refuse an act of type *offer* or *recommend*, but must follow the instruction of a *tell* act. When the content of an informative message is not computable, a response act of type *is_confused* is sent instead.

Complaining acts are used by the sender of a lower or an equal rank to seek the approval of a receiver for starting negotiation. COSMO does not define any complaining act for a sender of higher authority; such a sender can simply tell lower ranking receivers to start negotiation right away. Message types *notify* and *busy* can replace any of the response types for specific control purposes [Wong 92].

2.4 Communication and computation steps

A communicative act of one agent may spawn many other acts among agents; for instance, agent a asks b about p, b then asks c about p, and so on. A process of communication among agents is a **sequence** of communication and computation steps. A communication step encodes the specification of a sender into a message and sends it out to a receiver, i.e., performs a communicative act. A computation step changes or updates an agent's reasoning program, and when necessary, decodes an incoming message.

A sequence of these steps terminates properly when the first sender of that sequence receives a response message to its initial act. To keep track of multiple sequences occurring simultaneously in a cooperative system, the first message of any sequence contains a unique label, which will be included in all subsequent messages of the same sequence. In particular, we are interested in those sequences with precisely defined steps. We call such a sequence a **protocol**. Protocols are regulated means to convene complex intentions in terms of a few elementary communicative acts. They enforce an order on the way that cooperating agents interact with one another. An agent that wants to initiate communication of a specific aim would select a particular protocol. The agent would then communicate that it is using this protocol to the other agents. This would then allow the intended receivers to select corresponding acts and to ignore irrelevant messages. In this way, the protocol also avoids excessive communication and keeps discussion among agents under control.

We use the following format to describe communication and computation steps. For simplicity, we assume that every step in a protocol is an indivisible operation. A communication step whereby agent a invokes an act of type τ is expressed as: $a \rightarrow b \mid \tau$, p, l; where b is the receiver, the content p is a specification of a common format, and l is a unique identifier of the sequence. A computation step of agent a is written as $a \mid \pi(\mathrm{p})$, where π is an internal operation on the content p of an act in the previous step. The global language used in our prototypes is of a logical form such that p can be a term, a list, a predicate, or a complex sentence, i.e., predicates connected by logical operators such as $\wedge$ (and), $\vee$ (or), and $\supset$ (implication). The transformation operations of such a handler also include the parsing of a message content into its atomic components for evaluation in the reasoning program, and the composing of evaluated results into a proper logical form for reply.

In addition, there are many ways to compose protocols from a set of message types to solve the same problems. But one can compare the efficiency of these protocols by measuring the amount of time required for their computation and communication steps.

3 Communication Protocols

In COSMO, a multistage contract net protocol is used to allocate tasks in general [Conry 91], whereas a set of negotiation protocols, tied to particular decision-making methods, is used to resolve conflict of information among agents. Our experience shows that when the protocols become more sophisticated, they will be more closely related to the domain knowledge of the agents.

3.1 Preference-based negotiation

The set of negotiation protocols presented are used in conjunction with the group decision making methods in **social choice theory** [Sen 82]. In essence, these methods generally include an agenda that contains a list of criteria for each mutual problem. The agents would first form individual orderings of preferences on competing alternatives of a problem according to the specified criteria (which may vary for each agent). They then apply an aggregation procedure to select an outcome out of these individual preferences. Such an outcome is often expressed in the literature as a **collective choice**. Nevertheless, it is difficult to obtain a "fair" choice that satisfies all agents. Thus, in COSMO, we allow negotiation among agents to iron out the differences and uncertainties of individual preferences. In this regard, we depart from the social choice theory, which normally does not include the notion of **feedback** in the group decision making process.

The process of gathering and forming preference orderings in our scheme is briefly described as follows. An agent of the coordinator role would gather all individual orderings of preferences and combine them into one ordering according to a certain aggregation procedure. If any agent disagrees with the result, it can then protest to the coordinator in order to start a session of negotiation. This strategy allows only one coordinator in a cooperative system. The formation of individual preferences for cooperative problem solving is neither fixed nor arbitrary. In the context of knowledge base applications, individual preferences are normally derived from domain knowledge encoded in individual agents. A scheme based on the formalism of **preferential logic** that supports the derivation and aggregation of individual preferences for

O_a	O_b	O_c	O_g
z	x	z	z
x	y	y	x
y	z	x	y

Table 1. Individual ordering and aggregated ordering.

knowledge based applications has been presented in [Wong 92a]. The discussion of this scheme is beyond the scope of this paper, however, a simple example on aggregation is given below.

Let us suppose that there are three cooperating agents, a, b, and c, where a is the coordinator and b and c are peers, such that v(coordinator) > v(peer). Let us further suppose that each agent has its own criterion to judge its preferences, and the coordinator uses a standard procedure of aggregation: the simple majority rule. This rule specifies that, for a criterion, if both a and c prefer z to x while b alone prefers x to z, then the aggregated preference is that z is preferred to x, or $z > x$, with respect to that criterion. Table 1 shows the set of individual orderings O_a, O_b, and O_c, and the aggregated ordering of preferences O_g of a problem with competing alternatives x, y, and z. In this table, z is a collective choice as it is the highest ranking alternative in O_g. If there is a dispute, then the following strategy of negotiation is used.

1. Each of the agents computes a heuristic index by calculating the ranking difference of every feasible alternative between its individual ordering and the aggregated ordering.
2. Each of the agents checks if its index is over a threshold value, and, if so, asks the user whether it should complain or not; otherwise, it exits.
3. If any of the agents complains, start **bargaining**, and if bargaining fails, try **forcing**.

This negotiation strategy includes the users of individual agents in the decision making process. A variant of this strategy would be to have one or more agents to not ask local users. In Step 1, every agent first obtains a heuristic index of bargaining. For example, in Table 1, agent b calculates x's ranking difference between its ordering O_b and the aggregated ordering O_g as 1 and the total difference, that is, its heuristic index H_b, as 4. Similarly, we have $H_a = 0$ and $H_c = 4$. An agent uses its heuristic index to decide whether to flag the users for conflicts and to estimate whether the negotiation is converging towards a satisfactory solution.

(1)	$b \rightarrow a$	*protest*, protocol(bp_I), l
(2)	a	if approve then start bp_I; else tell b to abort
(3)	$a \rightarrow b$	*approve*, protocol(bp_I), l (assume approved)
(4)	b	determine to swap p (may exchange
		inform all agents about p (background knowledge)
(5)	$b \rightarrow a$	*recommend*, p, l
	$b \rightarrow a_1, ..., a_n$	*offer*, [p, protocol(bp_I)], l
(6)	$a, a_1, ..., a_n$	if p is feasible then accept; else reject
(7)	$a \rightarrow b$	*endorse/reject*, nil, l
	$a_i \rightarrow b$	*accept/refuse*, nil, l
(8)	b	if all agree then swap; else abort

Figure 2. The bargaining protocol bp_I of COSMO.

In Step 2, suppose that a uniform threshold, H_{th} = number of alternatives $- 1 = 2$, is applied across all agents. Thus, b and c both would query their users. Let us further suppose that the user of c decides not to complain, since z is in a reasonably good position in O_c. The user of b, however, would like to complain about the outcome as z is its least preferred choice.

The predominant mode of negotiating over a conflict in this strategy follows Galbraith's notion of **bargaining**, i.e., the agents push for acceptance of the alternative that is preferred by them and occasionally "give in" by making incremental changes to their preferred alternatives [Galbraith 77]. This treatment differs from the prevalent approach in distributed AI that presumes the knowledge of conflict resolution can be encoded and centralized in a special negotiator agent. COSMO uses several bargaining protocols. For brevity, this section illustrates three of them and focuses only on the discussion of agents' preferences.

3.2 Negotiation protocols

In Figure 2, we show a protocol of bargaining, bp_I. The protocol considers a set of agents $a, b, a_1, ..., a_n$ of the following role assignments: a is the coordinator and $b, a_1, ..., a_n$ are peers. Basically, any agent which wants to complain must first seek the approval of the coordinator. If approved, the agent would then attempt to persuade all other agents to swap two alternatives' positions in their individual orderings (computed in Step 4 as p), such that the expected aggregation results would favor its best choice. We skip

(1)	$b \rightarrow a$	*protest*, protocol(bp_I), l
(2)	a	start bp_I
(3)	$a \rightarrow b$	*approve*, protocol(bp_I), l
(4)	b	determine swap x, z
		inform all agents
(5)	$b \rightarrow a$	*recommend*, feasible(swap(a, x, z)), l
	$b \rightarrow c$	*offer*, [feasible(swap(c, x, z)), protocol(bp_I)], l
(6)	a	test that swap(a, x, z) is feasible
	c	swap(c, x, z) is feasible
(7)	$a \rightarrow b$	*endorse*, nil, l
	$c \rightarrow b$	*accept*, nil, l
(8)	b	invoke the change
(9)	$b \rightarrow c$	*offer*, swap(a, x, z), l
	$b \rightarrow a$	*recommend*, swap(c, x, z), l
(10)	$a \rightarrow b$	*endorse*, updated, l
	$c \rightarrow b$	*accept*, updated, l

Figure 3. A sequence of bargaining using Protocol bp_I.

O_a	O_b	O_c	O_g
x	x	x	x
z	y	y	y
y	z	z	z

Table 2. Preferences after the bargaining process in Figure 3.

the interchange of background knowledge among agents here. In Step 5, the protesting agent b also informs other agents about the protocol used so that the latter can select appropriate operations for this protocol. The protocol bp_I is successfully terminated only when all agents agree to the change.

Sometimes, the content of a response message is ignored when its type carries sufficient information. Step 7 of Figure 2 shows one such occasion, where the feasibility of p is indicated by the response types. Let us continue the example in Table 1. We show a sequence of steps using the bargaining protocol bp_I in Figure 3. In Step 5, feasible(swap(a,x,z)) denotes whether it is feasible to swap the positions of x and z in a's ordering. One of the conditions of feasibility is that the new index after the change should not be over the threshold value, i.e., 2. Other conditions involve checking the background knowledge that derives the preferences. This part concerns the

(1-5) is the same as (1-5) of Figure 9, except bp_II replaces bp_I

(6)	a	*test* that swap(a, x, z) is feasible
	c	swap(c, x, z) is infeasible
(7)	$a \rightarrow b$	*endorse*, nil, l
	$c \rightarrow b$	*refuse*, nil, l
(8)	b	not all agents agree, examine the would-be ordering (see Table 3) if okay then invoke change; else inform a, c to abort.
(9)	$b \rightarrow a$	*recommend*, abort(bp_II), l
	$b \rightarrow c$	*offer*, abort(bp_II), l
(10)	$a \rightarrow b$	*endorse*, nil, l
	$c \rightarrow b$	*accept*, nil, l

Figure 4. A sequence of bargaining using Protocol bp_II

underlying problem-solving methods of the reasoning programs, which will be discussed in a separate paper. Table 2 shows the new set of orderings after the completion of the bargaining in Figure 3. The set of heuristic indices are: $H_a = 2$, $H_b = 0$, and $H_c = 0$, where H_{th} is still 2.

Often it is difficult for all agents agree to a change. Consider the previous case, for c to change its best preference to the worst would require considerable revision of its domain knowledge, but b is not in a position to do so. Thus, the strategy of another bargaining protocol, bp_II, is to have an agent attempt to persuade, not all, but as many agents as possible to its preferred alternative, and hopefully, the new ordering of aggregation derived will be close to its expectation. One problem of this approach to bargaining is that an agent which refuses to change might stand to lose and, thus, would initiate another complaint. To avoid such situations, the protesting agent should also consider potential conflicts arising from its change. This is accomplished by the agent simulating the aggregation and detecting possible problems internally. For example, let us consider the case in Table 1 again. In Figure 4, agent b is still the protester that initiates bp_II, but this time c refuses to accept b's offer.

Table 3 shows why b decides not to press the change in Figure 4. The expected O_g: x > y > z would satisfy b, but would likely upset c as the latter would have an index of 4 and, thus, exceed the threshold value (H_{th}) of 2. Further, the 'best' collective choice x would be the worst choice from c's perspective. We consider that this kind of "good faith" negotiation, where the concerns of other agents are taken into account in negotiation, is essential

O_a	O_b	O_c	O_g
x	x	z	x
z	y	y	z
y	z	x	y

Table 3. Preferences after the bargaining process in Figure 4.

Old			New			Aggregate
O_a	O_b	O_c	O_a^{new}	O_b^{new}	O_c^{new}	O_g
z	x	z	x	x	y	x
x	y	y	z	z	x	z
y	z	x	y	y	z	y

Table 4. Preferences after the bargaining process using bp_III.

to cooperative behavior.

The two types of bargaining discussed so far are within a **fixed scope** of negotiation, that is, the agents are trying to reach an agreement under the same set of criteria in the problem agenda. Sometimes, the conflict may persist and no amount of such bargaining will provide for agreement. In COSMO, an agent may propose to use a third type of bargaining that proceeds to change the scope of negotiation. Instead of figuring out acceptable solutions under the existing problem agenda, this type of bargaining attempts to **reshape** that agenda, or rather, tries to replace the set of criteria with a different but closely related set. Changing the agenda is changing the subject of the conflict. Under the new criteria that the persuader brings up, it expects that the other agents would likely agree with its proposal. By getting the opposing agents to accept this new agenda, the persuader hopes to get them to join its side. In other words, we can say that this kind of bargaining is an attempt by one agent to change the opposing agents understanding of something, to get them to see it in some way (i.e., change their viewpoints) that prompts them to act as they would not have done otherwise.

The following example is used to illustrate such bargaining in COSMO. Consider agent b and Table 1 again. This time, b tries to persuade a and c to include a new criterion in the problem agenda. Referring to Figure 5, in Step 4, the term, new, is the additional criterion that b proposed, while reason(δ_{new}) is the justification of b to add this criterion. If any other agent is not satisfied with this justification, bargaining would be aborted here. This protocol succeeds when the coordinator agent, a, is ready to invoke the group

(1-3) is the same as (1-3) of Figure 3, except bp_III replaces bp_I

(4)	$b \rightarrow a$	*recommend*, [feasible(add_criterion(new)), reason(δ_{new})], l
	$b \rightarrow c$	*offer*, [feasible(add_criterion(new)), reason(δ_{new})], l

(5)	a, c	if reason(δ_{new}) is okay
		then accept add_criterion(new) is feasible;
		else reject.

(6)	$a \rightarrow b$	*endorse*, nil, l (assume both accept)
	$c \rightarrow b$	*accept*, nil, l
(7)	$b \rightarrow a$	*recommend*, [success(bp_III), add_criterion(new)], l
(8)	$a \rightarrow b$	*endorse*, nil, l
(9)	a	invoke contract net protocol to form new preference orderings.

Figure 5. A successful bargaining sequence of Protocol bp_III

decision making method to form a new set of preference orderings. Thus, the kind of bargaining used in bp_III is a **predecision** strategy.

Table 4 gives the results of new orderings. In this table, both old and new criteria (and thus the orderings) are taken to have equal importance in aggregation. The aggregated result shows that b gets what it wants out of this bargaining sequence, that is, to push for x as the collective choice.

Occasionally, **forcing** is used to back up the bargaining approach when a lack of agreement stymies the group. The coordinator can use the authority of its position to force a preferred alternative on the rest of the group. The protocol of forcing is invoked when the coordinator approves the request of an agent to break off the bargaining mode due to a lack of progress. In our applications, such a request to break off is triggered when the maximum number of offers allocated for the bargaining session is exceeded. Although forcing settles the problem of action so that the agents can progress, that is, the computation process can continue, it leaves the conflict unresolved. The agents are still at odds with their opposing beliefs about the matter as well as remaining in conflict. The use of forcing without proper direction from the domain knowledge would simply generate arbitrary solutions.

4 Conclusion

Cooperative problem solving will be conducted in many forms among a network of agents and will require the support of advanced communication facilities beyond the "passive" transmission of data and messages provided by the current network technology. CPS system developers should not be handicapped by the primitive concepts and constructs of existing communication schemes. Higher level concepts and constructs are needed so that developers are free to write the highly specialized parts that provide efficiency and are unique to a given application. This would help developers to focus on even-harder problems, pushing forward the state of the art and providing increasing value to the application users. In presenting this account, we attempt to offer new insight into the use of communicative acts and protocols to form advanced communication scheme for CPS systems.

謝辞 This work was supported in part by the NSF Engineering Research Center: Advanced Technology for Large Structural Systems (ATLSS) at Lehigh University, PA, USA, in part by the research grant STA-191101 from the Science and Technology Agency of Japan, and in part by NSF grant INT-9123128. The author would like to thank ICOT for their support in writing and presenting this work.

参考文献

[ATLSS 92] ATLSS Center. Sixth-year renewal proposal to the NSF, Vol. 2: projects, publications, and biosketches, ATLSS Drive, Lehigh University, Bethlehem, PA, 1992.

[Austin 56] J. L. Austin, *How to do things with words*, Oxford: Clarendon Press, 1956.

[Conry 91] S. E. Conry, R. A. Meyer, and V. R. Lesser, "Multistage negotiation for distributed satisfaction," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 21, No. 6, 1991, November/December, pp. 1462-1477.

[Galbraith 77] J. Galbraith, *Designing complex organizations*, Reading, Mass., Addison-Wesley, 1977.

[Searle 85] J. R. Searle and D. Vanderveken, D. *Foundation of illocutionary logic*, Cambridge University Press, 1985.

[Sen 82] A. Sen, *Choice, welfare, and measurement*, MIT Press, 1982.

[Wong 92] S. T. C. Wong, "COSMO: A communication scheme for cooperative knowledge-based systems," *IEEE Transactions on Systems, Man, and Cybernetics*, to appear, 1993.

[Wong 92a] S. T. C. Wong, "Preference-based decision making for cooperative knowledge-based systems," *ACM Transactions of Information Systems*, to appear, 1993.

[Wong 92b] S. T. C. Wong and J. L. Wilson, "A set of design guidelines for object-oriented deductive systems," *IEEE Transactions on Knowledge and Data Engineering*, to appear, 1993.