Conversation Pattern-based Anticipation of Teammates' Information Needs via Overhearing

Xiaocong Fan, John Yen
School of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
{zfan,jyen}@ist.psu.edu

Abstract

One research focus of human-centered teamwork is on advanced decision architectures that can help people make effective and timely decisions. This requires distributed team members to effectively establish shared situation awareness and to collaboratively develop explanations on how an unfamiliar situation might have been emerging. One key to achieve this goal is the ability to anticipate others' future information needs and to offer help proactively. In this paper we investigate a novel approach to anticipating teammates' information needs based on stepwise conversation pattern recognition, leveraging the idea of multi-party communication. This approach can be further extended to build a computational model for collaborative story building as needed in Recognition-Primed, naturalistic decision architectures.

1. Introduction

Human-centered teamwork, as a subdiscipline of multi-agent systems, has recently been gaining much attention including work on policy-based adjustable autonomy [19], collaboration strategies [18], and advanced decision architectures [6]. The RPD-agent architecture [6], which extends the CAST architecture [22] with a computational Recognition-Primed Decision (RPD) model, has been developed to support human-agent collaboration in developing shared situation awareness and in making decisions based on progressively refined recognitions.

The RPD [12] model focuses on recognizing the similarity between the current decision situation and previous experiences, which is claimed as the way how human experts make decisions in complex, dynamic, time-stress environment. The RPD model captures the cognitive activity undergoing in the mind of a decision maker when he/she faces a decision task. However, it becomes more interesting if a team of human experts can establish a shared mental model about the dynamic progress of the RPD process being pursued by the decision makers and offer timely assistance (e.g., information sharing) proactively. It is exactly under this vision that the *Collaborative-RPD* model [7] is proposed and the RPD-agent architecture [6] is implemented.

Exploring the collaborative nature of the RPD model introduces not only performance gains in team decision making but also the challenges inherent in collaborative computing. One challenge is how to effectively develop *shared situation awareness* among team members and to collaboratively build stories to explain how an unfamiliar situation might have been emerging and how it would evolve.

Obviously, to support story building in a dynamic, distributed environment, one key is the ability to anticipate other teammates' information needs and to provide the relevant information to the right party at the right time. Within the CAST system, we have investigated several approaches to anticipating others' information needs, including information-need anticipation based on team plan analysis [22], context-centric mechanism based on information-need graph reasoning [9], and indirect information needs reasoning [8]. However, all of the above approaches assume that agents in a team have common knowledge about the team process being pursued. This becomes a big limitation when there is a critical need for supporting 'unplanned collaboration', where a group of agents work together by constantly tracking each other's progress, by following certain social norms, protocols, or routine experiences. One intuitive approach is to enable agents to anticipate others information needs solely based on the social interactions, which, typically, are composed of conversation patterns.

Story building in teamwork settings typically involves multiple parties with various expertises. In this paper we investigate a novel approach to anticipating teammates' information needs based on stepwise conversation pattern recognition, leveraging the idea of multi-party communication.

This approach can be further extended to build a computational model for *collaborative story building*. The remainder of the paper is organized as follows. In Section 2, we review the state-of-art of multi-party communication. We introduce a formal representation model of multi-party conversation patterns and conversational experiences in Section 3, and give algorithms for pattern recognition and needs anticipation in Section 4. Section 5 concludes the paper.

2. Background: multi-party communication

Multi-party conversations are conversations involving more than two parties. Human society is full of examples of multi-party communication, such as panel discussions, presentations, and radio channels in the real world; net meeting, newsgroup, mailing lists, teleconference, and chat rooms in the digital world. Recently, research in the area of agent communication has been shifting attention to support multi-party conversations [14, 16, 11], motivated by the fact that the core of societal interactions is communicating within a group [14]. For instance, group communication is used to coordinate the role-taking of a group of agents who can play the same role [2]; Traum proposed a model of multi-party dialogue in immersive virtual worlds, mainly focusing on attentions and turn-takings [21].

Although multi-party communication can be implemented using the traditional multicasting or broadcasting technique, they are quite different. A multi-party communicative act can carry different intentional semantics to different parties, it is thus at a higher level than multicasting or broadcasting, just like the difference between the performative 'tell' and the physical 'send' operation. In addition, multi-party communication has to address issues such as recipients being unknown, the sender being a group, and the intended actors being different from the recipients [14].

Except for the blackboard metaphor [4, 5], probably the concept of *shared information net* implemented in DDD (Distributed Dynamic Decision-making simulation environment [13]) can be viewed as a prototype of multi-party communication. As illustrated in Fig. 1, human decision makers (agents) can 'register' themselves to as many pre-defined information nets (dotted ovals) as needed. An information net is a container of certain types of information dispatched from various sources (small ovals filled with colors), and all the decision makers who have established connectivity to an information net can access those information over the net. However, information nets have no support for information exchange among agents except for information sharing.

If we also view the information sources in Figure 1 as agents, the information nets become a richer concept– 'multicast channels' [1]. All messages exchanged on a channel can be heard by everybody tuned on the channel although some are not necessarily the intended audience. However, this by itself is not a new idea and has already been employed in the locker-room agreements–a communication protocol proposed for periodic team synchronization domains [20]. The novelty of channeled multicast is that it organizes streams of messages by channels with themes, and it supports a system development methodology where an agent society can be organized in terms of conversation roles and themed communication channels [3].

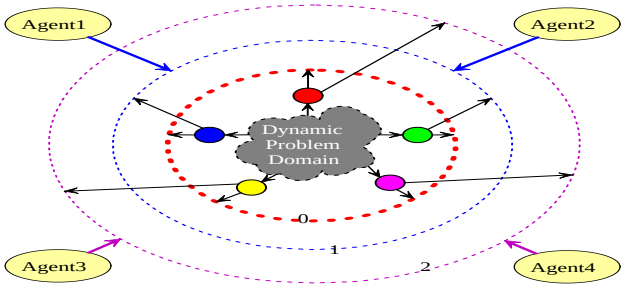

Figure 1. Multi-party communication

A multi-party dialogue involves more than two conversation roles, and the roles played by an agent can change during a dialogue [5]. In addition to the roles of speaker and addressee, the other roles include auditors (the intended listeners), overhearers (the anticipated but unintended listeners), and esvesdroppers (the unanticipated listeners). The existing work mostly focuses on the role of overhearer. In the overhearing architecture [3], overhearer agents are used to listen to one or more channels on behalf of subscribers, analyze the exchanged messages and selectively forward whatever information is relevant to the subscribers. Rossi and Busetta sketches a rule-based approach [17] where a group of agents by overhearing can conduct conversation state analysis and social role recognition, based on landmark-based representation of conversation protocols. Overhearing is also used for group formation [15] and in formally studying the conversation identification problem [10].

The benefits of multi-party communication are obvious: less communication cost, easier to establish mutual agreements (beliefs), better for enhancing social awareness of others' work progress and avoiding redundant assistance, enabling unobtrusive observations, unsolicited suggestions, and unplanned collaboration. In this paper, we specifically leverage multi-party communication for enabling agents to anticipate others' information needs based on stepwise conversation recognition. Gutnik and Kaminka [10] have formally studied the conversation recognition problem in an overhearing setting, and explored the complexity of the algorithms for handling lossless and lossy overhearing. However, for the purpose of our study, this work bears at least

two limitations. First, they assume the overhearers and the conversation participants share the same collection of conversation patterns. We drop this assumption—a group of RPD-agents can have different expertises (experiences). Second, it is a *static* approach to conversation recognition in the sense that the overhearer tries to determine a conversation pattern based on a *full sequence* of overheard messages. Quite differently, we believe a dynamic approach is more useful: if an overhearer can *stepwisely* recognize the pattern being used only based on the sequence of messages exchanged so far, then the overhearer might be able to satisfy the information needers before they actually initiate the ensuing conversations, and hence could successfully terminate the ongoing conversation beforehand. Moreover, it is an open issue in Gutnik and Kaminka's approach how to determine the start and the end of a conversation. This is no longer an issue if stepwise recognition is employed. In addition, Gutnik and Kaminka describe the joint conversation states as a product of the members' individual states. This is criticized [17] because the exact members of a communicating group are oftentimes unknown.

3. Representation of Conversation Patterns and Conversational Experiences

3.1. Conversation Patterns

We extend Petri-nets model to represent abstract multi-party conversation patterns.

Let Ω be a set of performatives. A multi-party conversation pattern π is a tuple $\langle P, T, F, \Upsilon, \Psi, \Gamma, \tau \rangle$, where

• P is a finite set of token places;

• T is a finite set of transitions;

• $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs connecting places and transitions;

• $\Upsilon = \Upsilon_0 \cup \{*\}$, where '$*$' represents the whole communicating group, Υ_0 is a finite set of (agent) role variables or role constraints. Variables in Υ_0 are either mutable or immutable. An immutable role variable retains its binding throughout a conversation after it is bound to an agent, while a mutable variable can take different bindings in its different occurrences. A mutable variable is represented in a bar notation (e.g., $\bar{m}$ is a mutable role variable). A role constraint in Υ_0 is represented as a first-order formula;

• Ψ is a finite set of variables for performative arguments;

• $\Gamma \subseteq T \times \Xi$, where Ξ is a set of transition labels of the form $\langle \rho, \nu_1, \cdots, \nu_k \rangle$, where $\rho \in \Omega$, $\nu_i \in \Psi (1 \leq i \leq k)$, k is the arity of the performative ρ. Variables $\nu_i (1 \leq i \leq k)$ will be replaced by domain-dependent wffs upon conversation instantiation. For example, $\langle ask, X \rangle$ is a transition label stating that the speaker is asking information about X. Given any $t \in T$, assume $\rho(t)$ returns the performative component of $\Gamma(t)$;

• $\tau \subseteq F \times \Upsilon$ is a set of arc labels. For any $t \in T$, let $.t = \{\langle p, t \rangle | p \in P, \langle p, t \rangle \in F\}$, $t. = \{\langle t, p \rangle | p \in P, \langle t, p \rangle \in F\}$. Let $\tau(.t) = \{\tau(\lambda) | \lambda \in .t\}$, $\tau(t.) = \{\tau(\lambda) | \lambda \in t.\}$. For any t, any $i \in \tau(.t)$, and $j \in \tau(t.)$, their combination represents a communication act from i to j of type $\rho(t)$.

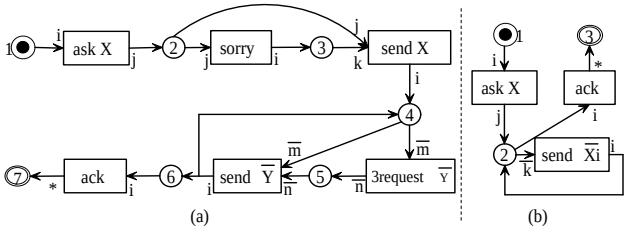

Figure 2. (a) Proactive-Response Pattern π_p, (b) Bundle-Ask Pattern π_b

Figure 2 gives two multi-party conversation patterns. The Proactive-Response pattern π_p in Figure 2(a) starts with an 'ask' from i to j about X. Then, j either sends X to i or says sorry followed by another agent k sends X to i. Next, any listener can infer the potential information needs relevant to X and then either sends the information to i or requests another agent to send the information to i. The pattern ends with an acknowledge from i to the whole group. The Bundle-Ask pattern π_b in Figure 2(b) also starts with an 'ask' from i to j about X, where X can be a conjunction expression. However, different agents, according to their expertise, may have different conjunction compositions for X. For example, from agent m's viewpoint, it could be $X = X_1 \wedge Y_2$, while from agent n's perspective, it could be $X = X_1 \wedge X_2 \wedge X_3$. Upon receiving an ask message, each group member will send whatever it has based on its decomposition of X until receiving an acknowledge from i. The bindings for mutable role variables (e.g., m and k) can be different for different iterations.

We assume performative arguments are represented by first-order expressions (predicates, conjunction). Let Φ be the set of first-order wffs regarding the domain problem under concern. An instantiation of a conversation pattern π with respect to a group of agents G is a tuple $\langle \pi, \varpi, \gamma, \upsilon \rangle$, where ϖ, γ, and υ give the current token configuration, the bindings for role variables in Υ_0 (valued over G), and the bindings for performative arguments (valued over Φ), respectively. For instance, $\varpi(\pi_p) = [0100000]$ means only place 2 of pattern π_p holds a token. A role variable can be bound to a subset of G, and $*$ is always bound to G, the whole communicating group. We use notation like $i : A$ to denote that i is bound to A; when i is a constant, it means i is replaced by A; it is a pointwise binding when both i and A are vectors.

Example: Agents A, B, C, and D as a group needs to monitor the airspace in the vicinity of an aircraft carrier, and to decide how to respond to the incoming air targets. Suppose C has the following knowledge:
$ThreatLevel(?a, High) \leftarrow Type(?a, Military) \wedge$
$ThreatBySD(?a, High) \wedge ThreatByAR(?a, High)$,
$Type(?a, Military) \leftarrow IFF(?a, > 1Mhz)$,
$ThreatBySD(?aHigh) \leftarrow Speed(?a, > 400mph) \wedge$
$Direction(?a, < 20deg) \wedge IFF(?a, > 1Mhz)$,
$ThreatByAR(?aHigh) \leftarrow Angle(?a, [-5deg, 5deg]) \wedge$
$Range(?a, < 80miles) \wedge IFF(?a, > 1Mhz)$.
Below is a possible conversation following the Proactive-Response pattern, where
the role mapping is $\{i : A, j : B, k : C, \bar{m} : C, \bar{n} : \overrightarrow{B, D}\}$,
the argument binding is $\{X : Type(a107, ?s)$,
$\bar{Y} : \overrightarrow{ThreatBySD(a107, ?t), ThreatByAR(a107, ?t)}\}$.
Here, we use $\bar{v} : \overrightarrow{X_1, X_2, \cdots}$ to represent that v is bound to the values of the sequence $X_1, X_2, \cdots$, one at a time.

A: Ask B, "tell me $Type(a107, ?s)$, I need it to judge $ThreatLevel(a107, High)$–whether there is a hight threat from the aircraft $a107$ "
B: Sorry, I don't know $Type(a107, ?s)$
C: Send A, "$Type(a107, Military)$"
C: Request B, "Send $ThreatBySD(a107, ?t)$ to A, cause A must be unaware of IFF, consequently A must need this information to judge $ThreatLevel(a107, High)$"
B: Send A, $ThreatBySD(a107, High)$
C: Request D, "Send $ThreatByAR(a107, ?t)$ to A, cause A must be unaware of IFF, consequently A must need this information to judge $ThreatLevel(a107, High)$"
D: Send A, $ThreatByAR(a107, High)$
A: Ack to all, "you are so helpful"

Such an abstract model offers us several benefits. First, the use of role variables and role bindings allows us to express a wide range of multi-party conversations. (a) If an 'in' arc and an 'out' arc of a transition are labeled with the same role variable, the transition can be used to express self-addressing messages such as "think aloud"; (b) If the role variable of an 'in' arc of a transition is bound to a set of agents, it can represent performatives with a group of initiators such as "We announce that ..."; (c) If an 'in' (or 'out') arc of a transition is labeled with a role constraint rather than a role variable, it can specify implicit conversation participants such as "whoever committed to task X needs to abandon it". Second, unlike the landmark-based representation of conversation, our model not only captures the types of performatives involved in a conversation, it also captures their temporal relations in terms of transition sequences. This enables us to design algorithms for recognizing the patterns being used and for offering proactive assistance regarding the information needs as the conversation further evolves. Third, the model allows us to explore a stepwise approach to conversation-pattern recognition, where both performative types and domain-specific information types are taken into account.

3.2. Conversational Experiences

An experience in the terminology of RPD [12] represents an agent's past response to a certain situation, consisting of four parts: relevant cues (what to pay attention to), plausible goals (which goals make sense), expectancy (what will happen next), and course of action (what actions worked in that situation).

We denote a conversational experience (i.e., an experience of conversation) e_i by a tuple $\langle C_i, E_i, G_i, A_i \rangle$, where
• C_i is a sequence of messages of form $\langle \theta_s, \theta_a, \rho, \nu_{[1..k]} \rangle$, where θ_s is the sender, θ_a is the addressee, ρ is the performative, and $\nu_{[1..k]}$ are the performative arguments. By default, the first element, $\nu[1]$, is the 'ontological' object of the performative. For instance, given message $\langle A, B, Ask, Loc(e2, ?x, ?y), C \rangle$, $\nu[1] = Loc(e2, ?x, ?y)$. Assume $\nu[1].type$ and $\nu[1].var$ return the type and the argument list of $\nu[1]$, respectively. For the above example, $\nu[1].type = Loc$, $\nu[1].var = (e2, ?x, ?y)$. We also use $\nu_{[2..]}$ to refer to the performative arguments excluding $\nu[1]$. Messages in C_i are cues to be used in "cue matching" to determine whether this experience is applicable to the current conversation situation;
• E_i is a set of expectancies of form $\langle \pi, \varpi \rangle$, where π refers to the conversation pattern that, upon instantiated, matches with C_i, and ϖ gives the token configuration of the conversation pattern after matching the sequence C_i. Hence, π, ϖ, and the variable bindings resulted from conversation matching together determine the expectancy with respect to π — the rest sequence of messages to be exchanged as the conversation further evolves;
• G_i is a set of goals denoted by first-order formulas. Typically, G_i is domain-dependent. For conversational experiences, G_i is null before a recognition (i.e., the process of finding similarity between the current situation and past experiences), populated with goals (e.g., $ThreatLevel(a107, High)$ in the example of last section) during a recognition, and passed to the recognizing agent after a recognition. G_i provides a *context* for anticipating the future information needs of teammates;
• A_i is a set of communicative acts for evolving the current conversation or for suggesting solutions to future collaboration opportunities (e.g., role allocation) based on the identified conversation pattern being used in the group. For example, an overhearer (say C in the last example) may send out a request on behalf of another team member, knowing that the team member itself would send out the same request as the situation further evolves. While E_i can be used

to derive teammates' information needs, A_i can be used to satisfy others' information needs; both are done in a proactive manner.

For example, here are two experiences of agent C:
$e_1 = \langle C_1, E_1, \emptyset, A_1 \rangle$, $e_2 = \langle C_2, E_2, \emptyset, A_2 \rangle$, where
$C_1 = [\langle \text{A}, \text{B}, Ask, Type(a107, ?s) \rangle]$,
$E_1 = \{\langle \pi_p, [0100000] \rangle, \langle \pi_b, [010] \rangle\}$,
$A_1 = \{\langle \text{C}, \text{B}, 3Request, ThreatBySD(a107, ?t), \text{A}, shareGround(Type, ThreatBySD) \rangle\}$
$C_2 = [\langle \text{A}, \text{B}, Ask, Type(a107, ?s) \rangle, \langle \text{C}, \text{A}, Send, Type(a107, Military) \rangle]$,
$E_2 = \{\langle \pi_p, [0001000] \rangle\}$,
$A_2 = \{\langle \text{C}, \text{D}, 3Request, ThreatByAR(a107, ?t), \text{A}, shareGround(Type, TreatByAR) \rangle\}$, where the third argument of $3Request$ is $shareGround(Type, ThreatByAR)\rangle\}$, which gives the reason of performing the third-party request: $Type$ and $ThreatByAR$ share the same ground (i.e., IFF).

Expectancy monitoring is one of the key features implemented in RPD-agent to support adaptive decision makings [6]. An expectancy states what will happen. An agent becomes more confident about its recognition (i.e., the identified experience once worked in a situation similar to the current one) when it obtains *positive* evidences that can validate the expectancies of the experience being considered. On the other hand, when an agent gets a *negative* evidence that invalidates an expectancy, the agent has to reconsider the current situation by either triggering a new recognizing process or refining the current recognition.

To enable recognition refinement, the RPD-agents use experience refinement relations to organize experiences about conversation patterns. For any experiences $e_i = \langle C_i, E_i, G_i, A_i \rangle$ and $e_j = \langle C_j, E_j, G_j, A_j \rangle$, e_i is *refined by* e_j, denoted by $e_i \sqsubseteq e_j$, iff $C_i \prec C_j$, where $\prec$ is a sequence prefix relation. Simply, an experience is a refinement of another if it considered more cues (typed information exchange) in the recognition. Continuing the above example, $e_1 \sqsubseteq e_2$ holds, because in e_2, more communicative acts were observed.

4. Needs Anticipation By Overhearing

Collaborative story building in the RPD process can benefit from multi-party communication in many ways. For instance, allowing communications to have multiple addressees and listeners (overhearers) can help team members sketch a local view of the development of the story being built, can help others link pieces of information using local expertise on information dependency, and can help others collect negative/positive evidence about the hypothesis being explored. In this paper, we only focus on the anticipation of others' information needs based on stepwise conversation pattern recognition.

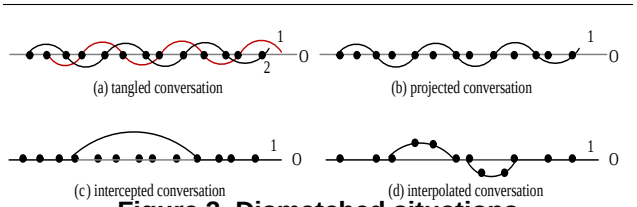

Figure 3. Dismatched situations

We assume the group of agents involved in a multi-party conversation (a) are cooperative, (b) have shared ontologies, and (c) obey the 'teleconference' style of communication (i.e., there can be at most one speaker at a time).

4.1. Stepwise Conversation Pattern Recognition

Instead of directly matching the observed message sequences with abstract conversation patterns, we explore an approach to match the observed message sequences with concrete conversational experiences, considering both performative types and domain-specific information types.

Below, by 'conversation' we mean a sequence of messages that is an instantiation of some (multi-party) conversation pattern. We use $O = o_1, \cdots, o_l$ to denote the sequence of observed messages, where $o_i (1 \leq i \leq l)$ is of form $\langle \theta_s, \theta_a, \rho, \nu_{[1..k]} \rangle$, where θ_s is the sender, θ_a is the addressee, ρ is the performative, and $\nu_{[1..k]}$ are the performative arguments. o_l is the latest message observed.

To match O with the cues C_i of an experience e_i, it is necessary to consider the following situations (see Fig. 3): (a) a sequence of observed messages can be the tangle of two or more conversations; (b) only the projection of the observed messages onto a subset of the communicating group can match with a conversational experience; (c) only a subsection of the observed messages can match with a conversational experience; and (d) the observed messages, after being interpolated with other messages, can match with a conversational experience.

The stepwise recognition algorithm given below has several distinct features. First, it is a backward-matching algorithm, starting from the last observed message (line 26). This avoids the difficulty of determining the first message in O that starts a new conversation. Second, it is a dynamic approach, allowing recognition refinement through monitoring the development of the expectancies of experiences in S_k (lines 19–25). This is different from the static matching considered by Gutnik and Kaminka [10]. Third, the algorithm covers the four cases illustrated in Fig. 3 by role-ignorance computing (line 3) and message-ignorance matching (lines 9 and 14). The role-ignorance computing complements Gutnik and Kaminka's lost-role analysis [10]: in our approach, messages in O from certain roles are virtually ignored in order to find a matching experience, while

Algorithm StepwiseRecognition
Input: O, //the sequence of observed messages
EKB, //the experience base
G, //the group of agents
$\top_\epsilon$, //preset threshold for workable deviation
$\bot_\epsilon$ //preset threshold for acceptable deviation
1. Let M be a list of subsets of G, ordered by set size decreasingly, where $M[0] = G$
2. For $k = 0..|M|$
3. O_k = O.ignore($G - M[k]$) //cases (a), (b)
4. S_k = **BackwardMatching**(EKB, O_k)
5. η = $min\{\delta_i | \langle e_i, \delta_i, \gamma_i, \upsilon_i, \{\pi^j, \varpi^j\}\rangle \in S_k\}$
6. If ($\eta \leq \bot_\epsilon$) Return(**NeedAnticipate**(S_k))
7. If ($\bot_\epsilon < \eta \leq \top_\epsilon$) Break //goto phase 2
8. If ($\eta \geq \top_\epsilon$)
9. S_k = **IgnoredMatch0**(EKB, O_k) //case (c)
10. η = $min\{\delta_i | \langle e_i, \delta_i, \gamma_i, \upsilon_i, \{\pi^j, \varpi^j\}\rangle \in S_k\}$
11. If ($\eta \leq \bot_\epsilon$) Return(**NeedAnticipate**(S_k))
12. If ($\bot_\epsilon < \eta \leq \top_\epsilon$) Break
13. If ($\eta \geq \top_\epsilon$)
14. S_k = **IgnoredMatch1**(EKB, O_k) //case (d)
15. η = $min\{\delta_i | \langle e_i, \delta_i, \gamma_i, \upsilon_i, \{\pi^j, \varpi^j\}\rangle \in S_k\}$
16. If ($\eta \leq \bot_\epsilon$) Return(**NeedAnticipate**(S_k))
17. If ($\bot_\epsilon < \eta \leq \top_\epsilon$) Break
18. end For
19η_1 = $min\{\delta_i | \langle e_i, \delta_i, \gamma_i, \upsilon_i, \{\pi^j, \varpi^j\}\rangle \in S_k\}$
20Repeat
21. $\eta_0 = \eta_1$
22. S_R = **RefineRecognition**(S_k, O') // new messages
23. η_1 = $min\{\delta_i | \langle e_i, \delta_i, \gamma_i, \upsilon_i, \{\pi^j, \varpi^j\}\rangle \in S_R\}$
24. If ($\eta_1 \leq \bot_\epsilon$) Return(**NeedAnticipate**(S_R))
25Until($\eta_0 < \eta_1$) //then, start a new recognition process

---

Algorithm BackwardMatching
Input: EKB, O
Output: $S = \{\langle e_i, \delta_i, \gamma_i, \upsilon_i, \{\pi^j, \varpi^j\}\rangle\}$
26. ES = MatchLast(EKB, o_l)
27. For each $e_i = \langle C_i, E_i, G_i, A_i\rangle \in ES$
28. count = 0; W=null; V=null
29. While (O.hasMore() & C_i.hasMore())
30. $\langle \theta_s^o, \theta_a^o, \rho^o, \nu^o\rangle = O$.getLast()
31. $\langle \theta_s^c, \theta_a^c, \rho^c, \nu^c\rangle = C_i$.getLast()
31. If ($\rho^o == \rho^c$)
32. $\gamma_i.addAll(\theta_s^c : \theta_s^o, \theta_a^c : \theta_a^o)$
33. If ($\nu^o[1].type == \nu^c[1].type$)
34. $\upsilon_i.addAll(\nu^c : \nu^o)$
35. else count = count +1
36. else count = count +1
37. end while
38. $\delta_i = \text{count}\ /(2 \cdot |O|)$
39. $S[e_i] = \langle e_i, \delta_i, \gamma_i, \upsilon_i, E_i\rangle$
40. end For
41. return S

in the lost-role analysis, messages from certain roles are actually not captured in O due to network errors or some other constraints. Message-ignorance matching is realized by IgnoredMatch0() and IgnoredMatch1(), which ignore certain messages in O and C_i (of an conversational experience), respectively. Forth, the algorithm is adjustable. The matching process is not run-to-completion, but controlled by two preset thresholds $\bot_\epsilon$ and $\top_\epsilon$. NeedAnticipate() is triggered whenever the least matching deviation η is no more than $\bot_\epsilon$; the control goes to the refinement phase whenever $\bot_\epsilon < \eta \leq \top_\epsilon$. Thus, the complexity of the algorithm depends on the values of $\bot_\epsilon$ and $\top_\epsilon$, as well as the set M, which in practice is much smaller than 2^G after being confined by heuristics.

The algorithms for IgnoredMatch0(), IgnoredMatch1(), and RefineRecognition() are omitted due to space limit.

4.2. Needs Anticipation

Information needs are domain related; to anticipate others' information needs requires domain-specific inference knowledge and the modeling of others' situation awareness.

Algorithm NeedAnticipate
Input: $S = \{\langle e_i, \delta_i, \gamma_i, \upsilon_i, \{\pi^j, \varpi^j\}\rangle\}$
1. For each $\langle e_i, \delta_i, \gamma_i, \upsilon_i, E_i\rangle \in S$
2. For each $\langle \pi^j, \varpi^j\rangle \in E_i$
3. PI= π^j.**ReifyPattern**($\varpi^j, \gamma_i, \upsilon_i$) //reified pattern
4. Let T_I be the subsequent transitions of PI wrt. ϖ^j
5. For each $t \in T_I$
6. $\langle \rho, \nu_{[1..k]}\rangle = \Gamma(t)$
7. If (ρ is 'ask') & (self.know($\nu[1]$))
8. self.send($\tau(.t), \nu[1]$)
9. If (ρ is '$send$') & (self.know($\nu[1]$))
10. self.send($\tau(t.), \nu[1]$)
11end For

NeedAnticipate() is only a skeleton algorithm for an overhearer to anticipate and help with other teammates' information needs. ReifyPattern() uses the bindings γ_i, υ_i, and the current token configuration ϖ^j to instantiate pattern π^j. PI is π^j under the context w.r.t. message sequence O; all the transition labels and arc labels in PI are replaced according to the bindings γ_i and υ_i. Lines 7-8 deal with ensuing 'ask': the overhearer replies the 'asker' with what it knows; Lines 9-10 deal with ensuing 'send': the overhearer provides the 'receiver' with what it knows. Codes can be added to deal with other cases like 'subscribe' and 'request'. In addition, the function know() in lines 7 and 9 can be extended to cover more complicated cases such as indirect information needs. For example, suppose agent A asks for K. An overhearer C can send K' to A, if it only knows K', which is indirectly related to K, say, by some inference rule.

5. Conclusion

Aiming for enhancing the capability of agents in proactive information delivery, in this paper, (1) we proposed an abstract model of multi-party conversations, where the use of role variables and role constraints allows us to express a wide range of conversation patterns; (2) we described a novel approach to anticipating teammates' future information needs based on stepwise conversation pattern recognition. The backward-matching algorithm is a dynamic approach; it allows recognition refinement through monitoring the development of the expectancies of the experiences under concern and considering newly observed messages.

While pattern recognition and anticipation can also be used in single-illocution communication, the novelty of our approach is that, by taking advantage of the idea of multi-party communication (especially the role of overhearer), it encourages 'joint' contributions in the process of recognition refinement and needs anticipation.

This is our first attempt of integrating multi-party communication into R-CAST [6]. Our next step is to empirically study the effectiveness of the algorithms on the developing of shared situation awareness and further to build a computational model for collaborative story building.

References

[1] P. Busetta, A. Dona, and M. Nori. Channeled multicast for group communications. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1280–1287, 2002.

[2] P. Busetta, M. Merzi, S. Rossi, and F. Legras. Intra-role coordination using group communication: A preliminary report. In *Workshop on Agent Communication Languages, LNAI 2922*, pages 231–253, 2003.

[3] P. Busetta, L. Serafini, D. Singh, and F. Zini. Extending multi-agent cooperation by overhearing. In *CooplS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*, pages 40–52. Springer-Verlag, 2001.

[4] P. R. Cohen, A. J. Cheyer, M. Wang, and S. C. Baeg. An open agent architecture. In *AAAI Spring Symposium*, pages 1–8, 1994.

[5] F. Dignum and G. Vreeswijk. Towards a testbed for multi-party dialogues. In *Workshop on Agent Communication Languages, LNAI 2922*, pages 212–230, 2003.

[6] X. Fan, S. Sun, M. McNeese, and J. Yen. Extending the recognition-primed decision model to support human-agent collaboration. In *AAMAS'05 (To Appear)*, July 2005.

[7] X. Fan, S. Sun, and J. Yen. On shared situation awareness for supporting human decision-making teams. In *2005 AAAI Spring Symposium on AI Technologies for Homeland Security*, pages 17–24, March 2005.

[8] X. Fan, R. Wang, B. Sun, S. Sun, and J. Yen. Multi-agent information dependence. In *Proceedings of the 2005 IEEE International conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 41–46, April 2005.

[9] X. Fan, J. Yen, R. Wang, S. Sun, and R. A. Volz. Context-Centric Proactive Information Delivery. In *Proceedings of the 2004 IEEE/WIC Intelligent Agent Technology conference*, pages 31–37. IEEE Press, October 2004.

[10] G. Gutnik and G. Kaminka. Towards a formal approach to overhearing: Algorithms for conversation identification. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 78–85, 2004.

[11] M. J. Huber, S. Kumar, P. R. Cohen, and D. R. McGee. A formal semantics for proxycommunicative acts. In J.-J. C. Meyer and M. Tambe, editors, *Intelligent Agents VIII (ATAL 2001)*, volume 2333 of *Lecture Notes in Computer Science*. Springer, 2002.

[12] G. A. Klein. The recognition-primed decision (rpd) model: Looking back, looking forward. In C. E. Zsambok and G. Klein, editors, *Naturalistic decision making*, pages 285–292. 1997.

[13] D. Kleinman, P. Young, and G. Higgins. The DDD-III: A tool for empirical research in adaptive organizations. In *Proceedings of the 1996 Command and Control Research and Technology Symposium*, 1996.

[14] S. Kumar, M. J. Huber, D. R. McGee, P. R. Cohen, and H. J. Levesque. Semantics of agent communication languages for group interaction. In *Proceedings of 17th National Conference on Artificial Intelligence (AAAI 2000)*, pages 42–47. AAAI Press/The MIT Press, July 2000.

[15] F. Legras and C. Tessier. Lotto: Group formation by overhearing in large teams. In F. Dignum, editor, *Advances in Agent Communication*, LNAI-2922, pages 254–270. Springer Verlag, 2003.

[16] J. Pitt, F. Guerin, and C. Stergiou. Protocols and intentional specifications of multi-party agent conversions for brokerage and auctions. In *Agents'2000*, pages 269–276, 2000.

[17] S. Rossi1 and P. Busetta. Towards monitoring of group interactions and social roles via overhearing. In M. Klusch, S. Ossowski, V. Kashyap, and et al, editors, *Cooperative Information Agents VIII: 8th International Workshop, CIA 2004*, LNCS-3191, pages 47 – 61. Springer Verlag, 2004.

[18] P. Scerri, D. V. Pynadath, L. Johnson, P. Rosenbloom, N. Schurr, and M. Tambe. A prototype infrastructure for distributed robot-agent-person teams. In *AAMAS'03*, pages 433–440, July 2003.

[19] M. Sierhuis, J. M. Bradshaw, A. Acquisti, R. van Hoof, R. Jeffers, and A. Uszok. Human-agent teamwork and adjustable autonomy in practice. In *7th International Symposium on Artificial Intelligence*, Nara, Japan, 2003.

[20] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *AI*, 110:241–273, 1999.

[21] D. Traum and J. Rickel. Embodied agents for multi-party dialogue in immersive virtual worlds. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 766–773, 2002.

[22] J. Yen, J. Yin, T. Ioerger, M. Miller, D. Xu, and R. Volz. Cast: Collaborative agents for simulating teamworks. In *Proceedings of IJCAI'2001*, pages 1135–1142, 2001.