TR-604

Intelligibility in Conversation

by
S. Motoike & H. Suzuki (Matsushita)

November, 1990

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

Intelligibility in Conversation

Sachiko MOTOIKE
Tokyo Information and Communications Research Laboratory
Matsushita Electric Industrial Co.,Ltd.
3-10-1, Higashi-mita, Tama-ku, Kawasaki 214 JAPAN

and

Hiroyuki SUZUKI
Tokyo Information and Communications Research Laboratory
Matsushita Electric Industrial Co.,Ltd.
3-10-1, Higashi-mita, Tama-ku, Kawasaki 214 JAPAN

Abstract

This paper considers what makes contents of a conversation in natural languages "intelligible" to all kinds of users of consultation systems. Users of the consultation systems have an interesting characteristic. The system cannot expect a user to understand all questions that are needed to be answered for solving his problem, even though he knows all the related facts, if those questions are not expressed in a proper way. It is one of the most difficult point to make an intelligible question to the user. By analyzing actual human consultation conversations, we define the intelligibility and propose a new algorithm for generating intelligible questions by using user models and the problem domain knowledge, followed by an implementation.

1. Introduction

This paper considers what makes contents of a conversation in natural languages easy to understand in consultation systems. A consultation is a conversation between two persons, a consultant and a consultee. Normally a consultation starts with the explanation of his problem by the consultee. Then the consultant makes some questions and the consultee follows them with simple answers or more detailed explanations of his problem. At the end the consultant shows a solution to the consultee. One of the most difficult point to build a system that can perform this kind of conversation is how to make questions intelligible to all consultees, *i.e.* users.

Currently, two methods are proposed concerning this "*intelligibility*". One method utilizes user's plans to make a smooth conversation. In such researches, (1) methods to recognize a user's plan, (2) methods to construct and extend the user model based on his plan, and (3) methods to reply to a user utilizing the user model have been considered. For example, Allen has proposed a method based on an analysis of user's intention to recognize user's plan [1]. Kaplan has proposed a method to detect user's belief which is presupposed by a user, and also has proposed another method to reply to a user indirectly utilizing this presupposition [6]. Carberry has been tackling a method to recognize user's plan [3], and has proposed a method for it using default inferences [4].

These researches commonly and implicitly assume that a user has his own plan for the goal of his conversation. However, we cannot assume that the user has his own plan for how to reach the goal of the consultation conversations. A user seldom has enough knowledge on the consultation

subject. This means that a smooth consultation should be achieved without relying on a user's plan.

Paris[8] and Moore and Swartout[7] has proposed other methods for a conversation with users who have low level knowledge of a conversation subject. Paris's system tailors a reply to a user's question according to the knowledge level of the user. Moore and Swartout's system tries to make an intelligible reply to a user's question utilizing both structural knowledge of a conversation domain and dialogue context. However, both Paris's system and Moore and Swartout's system only reply to the questions users make. Their methods are not directly applicable to make intelligible questions which are needed in building consultation systems.

There is another problem to build consultation systems which the above researches have not dealt with, that is, a strategy to extract necessary information from the user smoothly and speedy.

This paper proposes a new framework which allows a system to perform a smooth and intelligible consultation, which is effective to solve the user's problem. In order to introduce "intelligibility" into a conversation system, we utilize information and knowledge which relate to the consultation goal, and which a user understands and maintains. [1]

In the following sections, a method to bring intelligibility into a conversation is proposed. Section 2 defines intelligibility and section 3 shows information and knowledge for a consultation goal in order to introduce intelligibility into a conversation. Section 4 explains a mechanism to realize intelligibility.

2. Intelligibility

In this section, we try to define the concept of intelligibility by observing and analyzing intelligible consultations about making patent licensing contracts. Figure 1 shows an actual, originally in Japanese, conversation between two persons. The consultee is an absolute novice in making a contract and does not have his own plan. The consultant, an expert of a consultation about contracts, controls the conversation smoothly without using any technical term. However, at the same time, he succeeded in extracting enough information to make the contract.

See the utterance E03 of Figure 1. It is a question issued from the consultant in order to get information who the licensor[2] is in this contract. Since objects such as "licensor" and "licensee" are technical terms, the consultee, a novice, is not familiar with these words and may not understand the direct questions like "who is the licensor?" However, he should know who owns the patents and understands the question "who owns those patents?" which is sufficient to obtain the name of the licensor. This is possible because the consultant knows two things : that the consultee has the information of a related matter to the fact that who owns patents, and that the owner of those patents is equal to the licensor. E05 is also of this kind of questions. The consultant who would like to know what is the licensed product in the contract should ask "for what products you use them(the patents)" by utilizing a relation "using patents for some products".

The second important point appears at the utterance E09. The consultant seems to utter E09 supposing that the following constraint relation is known to the consultee : a party who uses patents of another party produces and sells some products. The reason for this is :

> If the consultee has this constraint relation in his mind, he may apply it to his own utterance N02, and he may already be conscious of the result of this application, that is,

[1] Though a user may not have a definite plan for how to reach the goal, he should know the final goal of a consultation. The fact that he comes to consult implies that he knows the final goal.

[2] A licensor is a party who gets compensation from another party. A licensee is an opposite to a licensor. In this case, the consultee is a licensee.

E01: Hello, what's happened to you?
N02: We use patents for our products.
E03: Who owns those patents?
N04: Company G.
E05: For what products you use them?
N06: A heater of blanket.
E07: How many patents of Company G you use now?
N08: Now, only two. We've known of the fact already, and the other company has been claimed from Company G because of using Company G's patents.
E09: So, you've produced and sold those products, haven't you?
N10: Yes.
E11: How long, and how much?
N12: Yes, for about 5 years.
E13: About 5 years, maybe you have sold a lot!
N14: Yes. But I don't know its quantity.
E15: You have to pay for the past release, OK?
And.... Company G probably has other patents, doesn't it?
N16: No, it doesn't have any other patents for that product, at least now.
E17: At least now.....
.....

Nb: (E*, N* denote utterances of the consultant and the consultee, respectively)

Figure 1: Dialogue Example 1, for a Novice Consultee

the consultee's having produced and sold some products, or he may be able to conscious of it. Therefore, the consultant is able to confirm *positively* as "so, you've produced and sold those products, haven't you?" Oppositely, if the consultee didn't have this constraint relation as his knowledge, he may not have understood relationship between his utterance N02 and a state of affair of his having produced and sold those products. In this case, the consultant would be ready to ask the consultee *negatively* as "I think maybe you've produced and sold those products. Is it right?", because the consultant would take care not to make conversation go beyond the consultee's knowledge level and perform the conversation smoothly.

The third observation covers many parts of the conversation. The consultant supposes that the consultee maintains information related to the contents of consultant's queries. In general, when asking some matter to a conversation partner, the consultant can make more smooth conversation by considering whether the matter is known to the partner or not. For example, when a questioner doesn't know whether a partner has a watch or not, it is better to ask "do you know what time it is?" than to ask "what time is it?". If the partner does not have a watch and is asked "what time is it?", he cannot directly answer to the question. This gives a damage to the smoothness of the conversation. By asking "do you know what time it is?", the questioner can keep the partner away from a situation in which the partner cannot answer to the question. In Figure 1's case the consultant supposes that many kinds of information are known to the consultee. This depends on the fact that the consultee utters that he himself uses the patents, and he uses them for some products at the beginning of the conversation. Without such utterances, the consultant ought not suppose that the consultee might have more information than he has already uttered.

E21: Please tell me about the licensee.
N22: Licensee is Matsushita Electric Industrial Co., Ltd., and its location is the .....
E23: Then, next about the licensor, please.
N24: It is Company G. I don't know its location.

.....

Nb: (E*, N* denote utterances of the consultant and the consultee, respectively)

Figure 2: Dialogue Example 2, for an Expert Consultee

Two insights are gained from these observations. The first is that the consultant is able to extract necessary information without using any technical term. The consultant changes his viewpoint and uses information and knowledge which are represented only by objects and relations easy to understand for the consultee. The second is that the consultant must use appropriate objects and relations, and those only if possible, which are ready to be introduced to the conversation.

The easiness and appropriateness of constituents of a conversation is the main factor of *intelligibility*. Intelligible objects and relations, and constraint relations offer intelligible topics, and intelligible topic contributes to a smooth conversation.

Another conversation shown in Figure 2 is also a consultation of making a patent licensing contract. This conversation differs from the first one in the point that the consultee has some knowledge about making a contract. In this case, the consultant performs a conversation differently. For example, the consultant directly asks "who is a licensee?", and "who is a licensor?", etc. However, the consultant still maintains another kind of *intelligibility* for the consultee as well as in the Figure 1. It is easily-understandable for this consultee and is appropriate to perform a conversation using technical terms, since he has fairly high level knowledge and may have some plan of making a contract.

It becomes clear from these observations that intelligibility is deeply concerned with relationships between constituents of a conversation and a knowledge level of a consultee. It is necessary to have a framework to express the knowledge level of a user, in order to bring intelligibility into consultation systems.

Here we define "*intelligibility*" as follows : for some interlocutor constituents of a conversation, such as objects and relations, and constraint relations, are intelligible if and only if that interlocutor understands those meaning and usage, and is able to utilize them in a conversation appropriately without any difficulty.

3. How to Bring Intelligibility into Conversation Systems

This section explains knowledge and information which a consultation system must prepare in order to reply intelligibly to its user, and the method to reply utilizing them.

First, the system needs knowledge what kind of user the current user is, *i.e.* a user is typified by this knowledge. Once he is typified to a particular knowledge level, the system grasps which objects, relations, and constraint relations are easy to understand for him, and which constraint relations he attunes to. This knowledge level is called a *user type*. Second, it is necessary for the system to have a temporal state of a user. This is a chunk of information which the user is thought to keep at some time point of a conversation. We call this information chunk as a *user model*. Both user types and user models are used to reply to him intelligibly.

To represent a user's temporal problem state in a consultation, knowledge and information

other than user types and user models are necessary. We call these knowledge and information, *problem types* and *problem models*. A problem type corresponds to system's knowledge necessary to solve a user's problem. A problem model is a chunk of information which the system considers to be necessary to solve this problem.

Many systems perform a conversation appropriately, although they don't separate knowledge and information of a user from knowledge and information necessary for problem solving. This is partly because they deal with users who have their own plan. However, the separation is necessary in consultation systems to overcome one of the major problems in consultation systems - to extract necessary information for problem solving from its user in an appropriate way. Introducing user types and user models also relates to this problem. Introducing problem types and problem models has a relation to problem solving techniques.

In the following subsections, user types and user models are defined and the method to bring intelligibility into a conversation is proposed.

3.1. User type

An user type is the system's knowledge about what kind of user a current user is. Constituents of user type are:

(1) Qualifying conditions,

(2) Objects and relations which are easy to understand for a user,

(3) Constraint relations which are easy to understand for a user, and

(4) Constraint relations to which a user attunes.

Figure 3 shows an example of user type. This example is written in the notation of soa(state of affairs) with labels [5, 10]. [3] This user type typifies a user who is a licensee but a novice about making a patent licensing contract.

(a) corresponds to '(1) qualifying conditions. ' Qualifying conditions are conditions whether a user is typified by this user type or not. They are used when the system selects acceptable user types for the current user. The selection mechanism is explained later in section 3.3.

'(2) Objects and relations which are easy understand for a user' corresponds to (b) and (c) in Figure 3. This piece of knowledge is used to enable the system to keep intelligibility by using these objects and relations for a user of this user type. The content (b) says that objects and relations related to "producing" are intelligible to a user of this type. The system is allowed to give such topics that are expressed in a relation "producing", objects "agent who produces", "products which are produced", "places where agent produces products", "time when agent produces products".

'(3) Constraint relations which are easy to understand for a user' corresponds to (d) in Figure 3. The system is able to perform a conversation supposing that the user of this type understands this constraint relation. (d) says that a user easily understands a constraint relation that a party who produces products sells them. This allows the system which obtains information that the party X produces products somewhere to apply (d) to it and obtain new information that the party X sells those products somewhere. As a result, when asking a question related to this new information, the system may suppose that the user knows this constraint relation. For example, the system is able to ask positively to the user "so, you have sold those products, haven't you?

[3] We uses the situation theoretic notions and notations. See [2].

(a) "user is a licensee"
《being_a_licensee,object:"user";1》

(b) "user understands objects and relation concerning "producing""
《know,agent:"user",
object:《produce,agent:X,product:Y, place:Z,time:W;1》;1》

(c) "user understands objects and relation concerning "selling""
《know,agent:"user",
object:《sell,agent:X,product:Y, place:Z,time:W;1》;1》

(d) "user understands constraint about "a party who produces products sells them""
《know,agent:"user",object:《implies,
rel1:《produce,agent:X,product:Y, place:Z1,time:W;1》,
rel2:《sell,agent:X,product:Y, place:Z2,time:W;1》;1》;1》

(e) constraint1:"if user knows of a fact that he produces products.
then he also knows of a fact that he sells them"
《implies,
rel1:《know,agent:"user", object:《produce,agent:"user",product:Y,
place:Z1,time:W;1》;1》,
rel2:《know,agent:"user", object:《sell,agent:"user",product:Y,
place:Z2,time:W;1》;1》;1》

Figure 3: An Example of User Type, for a Novice Licensee

Where have you sold them?" [4] If the current user is not of this user type, the system would either simply ask "Where have you sold those products?" ignoring such a constraint relation, or would tell all the related things and ask "If you have produced those products somewhere, then you might also have sold them somewhere. Where have you sold them?"

From (2) and (3), a semi-ordered relation of user types is derived. A particular user type U1 is higher than another user type U2, when contents of (2) and (3) of U1 include all the contents of (2) and (3) of U2. This relation is utilized in section 3.3.

'(4) Constraint relations to which a user attunes' is a different kind of knowledge from (2) and (3). (2) and (3) are the system's belief about knowledge which the user keeps in his mind, but (4) is the system's belief about information which the user maintains temporally.

(e) in Figure 3 corresponds to (4). It shows constraint relations for extending user model as explained in section 3.2. Some constraint relations exist between two states, a state that someone has some information, and a state that he has another information. Content (e) means that if

[4] Some inappropriate cases might occur when asking this kind of questions. The latter part of the system's utterance is performed supposing that the user maintains information about the places where the user would sell products. A method to remove this inappropriateness is explained in section 3.2.

a user of this type has information being related to his producing then he also has information being related to his selling. In case a user attunes to this constraint relation, after the system obtains information that he has information being related to his producing, the system would be able to advance a conversation supposing that he also has information being related to his selling. Details are explained in section 3.2.

3.2. User model

A user model is a system's belief about the state of the user. Consequently, it changes as the conversation proceeds. It consists of two sorts of the system's belief.

The first one is a chunk of information which a user has given to the system by his utterance. The system may regard a user's utterance as a part of whole information which he keeps in his mind.

The second one is a chunk of information which the system believes this user is able to give. This type of information comes from the result of applying constraint relations to which this user attunes, which is described in a user type. This allows the system to continue the conversation supposing that the user keeps this kind of information.

Figure 4 shows an example of a user model. A user model is a set of soa's with labels. This example shows a user model just after the user uttered that he has used patents.

User models are utilized in two phases for selecting intelligible topics. The first phase is to check acceptability of a current user type. This will be explained in section 3.3. The second phase is to know whether or not the system is able to ask a particular topic supposing that a user maintains information related to this topic. Even if a person is able to understand a relation, he might not maintain information related to the relation. People retain the conversation smoothness by being conscious of this fundamental fact. The system has to properly suppose which information the user really maintains.

There are two cases in which the system is able to suppose that a user has information being related to the system's query. The first case is when a content of its utterance is related to a matter which a user really uttered before. The system may perform its utterance upon this supposition. An example is shown in Figure 4. When the system wants to ask more information about patents, the system is able to ask as "tell me about the patents. " This is possible because this user model shows that the user has information about the facts that the user uses the patents. The second case is when a user type has a constraint relation. For example, when the constraint relation, if the user knows the fact that he uses patents, then he also knows how many patents he uses, is in the user type of the current user, the system may extend the user model and know that the user knows how many patents he uses. This allows the system to ask as "how many patents do you use?", when the system wants to ask information about a number of the patents.

Conversely, there are two cases in which it is appropriate for the system to ask a question in an indirect way like "do you know how many patents you use?" The first case is when the

⟪use_patent,agent:"user",patent:patent1, time:time2;1⟫ &
⟪overlaps,time1:time1,time2:"now";1⟫ &
⟪being a patent,object:patent1,number:string3, country:country4;1⟫ &
⟪being_a_string,object:string3;1⟫ &
⟪being_a_time,object:time2;1⟫ &

Figure 4: An Example of User Model

user type doesn't contain the constraint relation. The second case is that when the system is not able to extend the user model with the constraint relation due to the lack of information if the user uses patents. In these two cases the user model does not guarantee that the user knows how many patents he uses. Hence it is more appropriate to ask "do you know how many patents you use?"

3.3. Decision of User Type

The system checks acceptability of a current user type, selects appropriate one by using a user model and a problem model. The system selects an appropriate user type by the following process every time the user utters.

1. *Listing candidates of the current user type*
   The system checks qualifying conditions of all the user types. (a) in Figure 3 is an example of the qualifying condition. The condition says that a user of this type must be a licensee. The system refers a problem model and checks if the condition is satisfied. One of the three cases occurs after this check, *i.e.* the condition is 'right', 'wrong' and 'unknown'. The last one means that any condition is not applicable. Candidates are the user types whose qualifying conditions are 'right' or 'unknown'.

2. *Selecting one user type from candidates*
   Since a user model includes all the information which the user has given to the system, all the objects and relations, and constraint relations in a user model are easy to understand for the user. The system selects a user type whose objects and relations, and constraint relations are included in the user model. A user type selected in this way is the maximum lower bound in the semi-ordered user types by knowledge level of a user.

This semi-ordered relation is represented hereafter by the sign $>$. This sign is assumed to have the following characteristics :

> for two user types $U1$ and $U2$, relation $U1 > U2$ holds iff all the objects and relations, constraint relations in $U1$ includes all those in $U2$.

Call this state that $U1$ is upper of $U2$. Suppose U is a set of candidates of user type, $U1$ and $U2$ are elements of U and $U1 > U2$, and any other user type $U3$ in U doesn't satisfy $U1 > U3$ or $U3 > U2$. In this case, if all the objects, relations and constraint relations in the current user model are included in any V in $\{u|u > U1 \ \ or \ \ u = U1\}$, and includes any V in $\{u|U2 > u \ \ or \ \ u = U2\}$, then $U2$ is the maximum lower bound.

The system performs a conversation very close to the user's knowledge level by selecting a user type according to the above process.

After selecting a user type, the system performs some actions if a different user type is preferred.

First, there are two cases which are concerned with acceptability check of user types.

- *The result of acceptability check changes 'right' to 'wrong'*
  If the result of acceptability check changes 'right' to 'wrong' at an earlier stage of a conversation, the system considers that a problem model has been constructed wrong, exchanges an old user type for the right one, and reconstructs a user model using the new user type and conversation history. In later stages, the system asks the user if his utterance is acceptable in this conversation.

- *The result of acceptability check changes 'unknown' to 'wrong'*
  First, the system exchanges the old user type for the new one. After that the system performs one of the following actions. In the case that a new user type is upper of the old user type, the system re-extends the user model with this new user type. In other cases, it reconstructs the user model using this new type and conversation history.

Second, in case that the system comes to know that the user has higher knowledge level than the system supposes before, the system exchanges the old lower user type to a new upper one, and re-extends a user model with this new one.

It is unnecessary to consider a case that the system comes to know the user has lower knowledge level than the system supposes, because the system always selects a lower level type than the user model, as shown in the selecting processes.

4. Selection of an intelligible topic through dialogue

A conversation system, *ToR*, has been implemented in ESP(Extended Self-contained Prolog) on PSI(Personal Sequential Inference Machine) [9]. This section explains how the actual consultation system selects an intelligible topic.

The final goal of ToR is to solve a user's problem, making a patent licensing contract. This goal equals a state that a problem model is typified by a particular problem type acceptably, and necessary information for making a contract is extracted in the problem model. ToR proceeds a consultation towards this goal, selecting appropriate topics being based on the framework of a user type and a user model.

In case that ToR has already determined a problem type[5] and a user type, it selects an intelligible topic to the user as follows :

1. *Search for a soa which has to be obtained by comparing the problem model and the problem type*
2. *Judge by the user type whether the objects and relations which expresses the above soa is intelligible to the user or not. In case that they are not intelligible to the user, it finds another soa which may give equal information as the soa above*
3. *Decide how to show the soa to the user*

Figure 5 explains ToR's action. It is supposed here that the user is a novice on making patent licensing contracts in which he shall be a licensee. It is also supposed that ToR has selected the user type shown in Figure 3, and the problem type shown in Figure 6.

Figure 5 shows a particular phase of the conversation in which the user utters "we produce products in the U.S. and Japan," to ToR's query "where do you produce them?" From the background of this utterance the user model UM and the problem model PM come to have contents written in state1 as their parts. After applying constraints, UM and PM get results written in state2. ToR applies constraint1 in Figure 3 to UM, and constraint2, constraint3, and constraint4 in Figure 6 to PM, respectively.

In next three stages, ToR extends UM and PM, then it starts to consider what to say next. In the first stage, by comparing the problem type and PM ToR searches for a soa related to information which is needed to make a contract. Figure 6 shows that ToR wants to obtain two soas, that is ⟪being_a producing_place,object:place2;1⟫ and ⟪being_a_selling_place,object:place4;1⟫, and place2

[5]The decision of a problem type is performed slightly different from a user type.

```
at state1:
    in UM:
        ⟪know,agent:"user",
            theme:⟪produce,agent:"user",product:product1, place:place2,time:time3;1⟫;1⟫
    in PM:
        ⟪produce,agent:"user",product:product1, place:place2,time:time3;1⟫&
        ⟪element_of,set:place2,element:"united_states";1⟫ &
        ⟪element_of,set:place2,element:"japan";1⟫
at state2:

    in UM:
        ⟪know,agent:"user",
            theme:⟪produce,agent:"user",product:product1,place:place2,
                    time:time3;1⟫;1⟫ &
        ⟪know,agent:"user",
            theme:⟪sell,agent:"user",product:product1, place:place4,time:time3;1⟫;1⟫
    in PM:
        ⟪produce,agent:"user",product:product1, place:place2,time:time3;1⟫ &
        ⟪element_of,set:place2,element:"united_states";1⟫ &
        ⟪element_of,set:place2,element:"japan";1⟫ &
        ⟪sell,agent:"user",product:product1, place:place4,time:time3;1⟫ &
        ⟪being_a_producing_place,object:place2;1⟫ &
        ⟪being_a_selling_place,object:place4;1⟫
```

Figure 5: An Example of User Model and Problem Model in Two States

and place4 have to be a set of countries. At this time, PM already has the former soa in it satisfyingly, but doesn't have the latter one, because place4 is not written as a set of countries. As a result, ToR wants to obtain a soa1 ⟪being_a_selling_place,object:place4;1⟫ in which place4 is a set of countries.

In the second stage, ToR checks the appropriateness of the selected soa. From the user type in Figure 3, this relation "*being_a_selling place*" is not easy to understand for the user, therefore, ToR tries to find another soa that includes place4 from PM. There exists ⟪sell,agent:"user",product:product1, place:place4;1⟫ in PM. Now ToR intends to obtain a soa ⟪sell,agent:"user",product:product1, place:place4;1⟫ in which place4 is a set of countries, and checks its appropriateness. A relation "*sell* " is easy to understand for this user, therefore, ToR is able to use this relation in the next topic.

In the third stage, ToR decides how to ask it for the user. ToR knows from UM that the user has information about a soa, ⟪sell,agent:"user",product:product1, place:place4;1⟫. This shows that it is not problematic to ask it directly to the user, namely, supposing that the user has some information about this soa. ToR also knows from the user type that the user understands a constraint relation that "if someone produces products then he sells them", consequently, ToR is able to ask it for the user supposing that the user has knowledge of this constraint.

After these considerations, ToR asks the user as "so, you sell those products somewhere, don't you? In which countries do you sell them?" As this example shows, by referring and comparing a user type, a user model, a problem type and a problem model, ToR searches for a soa related to necessary information, tunes it to an appropriate topic and as a result performs an intelligible

"System needs a soa that expresses licensed producing places, and it has to satisfy restrictions on parameters in it"
《need,agent:"system",
soa:《being_a_producing_place, object:A@set(country);1》;1》

"System needs a soa that expresses licensed selling places, and it has to satisfy restrictions on parameters in it"
《need,agent:"system",
soa:《being_a_selling_place, object:A@set(country);1》;1》

constraint2:"if someone produces products then he sells them, too"
《implies,
rel1:《produce,agent:X,product:Y, place:Z1,time:W;1》,
rel2:《sell,agent:X,product:Y, place:Z2,time:W;1》;1》

constraint3:"places where a licensee produces products can be regarded as licensed producing places"
《implies,
rel1:《produce,agent:X,product:Y, place:Z,time:W;1》 &
《being_a_licensee,object:X;1》 &
《being_a_licensed_product,object:Y;1》,
rel2.《being a producing_place,object:Z;1》;1》

constraint4:"places where a licensee sells products can be regarded as licensed selling places"
《implies,
rel1:《sell,agent:X,product:Y, place:Z,time:W;1》 &
《being_a_licensee,object:X;1》 &
《being_a_licensed_product,object:Y;1》,
rel2:《being_a_selling_place,object:Z;1》;1》

Figure 6: An Example of Problem Type, for a Licensee Using Licensor's Patents

conversation to the user.

5. Conclusion

A new framework for incorporating intelligibility into consultation systems is proposed. The framework utilizes knowledge and information about its user, and knowledge and information about user's problem to select appropriate words and topics for making a consultation conversation intelligible. ToR, a consultation system for making a patent licensing contract, is implemented based on this framework.

Acknowledgment

The authors would like to thank all our colleques, especially Katsura Kawakami and Yosimasa Goto, for their valuable comments and warm advices. This work is partially supported by the Institute for New Generation Computer Technology(ICOT) under contract S4303.

References

1. J. Allen, Recognizing Intentions From Natural Language Utterances, in *Computational Models of Discourse*, eds. M. Brady and R.C. Berwick, (MIT Press, 1983), pp.107-166.

2. J. Barwise and J. Perry, *Situations and Attitudes*, (MIT Press, 1983).

3. S. Carberry, Modeling the User's Plans and Goals, in *Computational Linguistics*. **Vol.14, No.3** (1988), pp.23-37.

4. S. Carberry, A Model of Plan Recognition that Facilitates Default Inferences, in *Proceedings of the Second International Workshop on User Modeling*, (1990).

5. J. Barwise and R. Cooper, Generalized Quantifiers in Situation Semantics, in preparation.

6. S.J. Kaplan, Cooperative Responses From a Portable Natural Language Database Query System, in *Computational Models of Discourse*, eds. M. Brady and R.C. Berwick, (MIT Press, 1983), pp.167-208.

7. Johanna D. Moore and William R. Swartout, A reactive approach to explanation, in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, (1989). pp.1504-1510.

8. C.L. Paris, Tailoring Object Descriptions to a User's Level of Expertise, in *Computational Linguistics*, **Vol.14, No.3** (1988), pp.64-78.

9. ICOT, Conversation Management System based on Contextual Understanding. in *ICOT Annual Report*, (1988).

10. H. Yasukawa, H. Suzuki and N. Noguchi, Knowledge Representation Language based on Situation Theory, in *Programming of Future Generation Computers II*, eds. K. Fuchi and L. Kott. (Elsevier Science Publishers B.B., North-Holland,1988), pp.431-460.

References

[1] J. Allen, Recognizing Intentions From Natural Language Utterances, in *Computational Models of Discourse*, eds. M. Brady and R.C. Berwick, (MIT Press, 1983), pp.107-166.

[2] J. Barwise and J. Perry, *Situations and Attitudes*, (MIT Press, 1983).

[3] S. Carberry, Modeling the User's Plans and Goals, in *Computational Linguistics*, **Vol.14, No.3** (1988), pp.23-37.

[4] S. Carberry, A Model of Plan Recognition that Facilitates Default Inferences, in *Proceedings of the Second International Workshop on User Modeling*, (1990).

[5] J. Barwise and R. Cooper, Generalized Quantifiers in Situation Semantics, in preparation.

[6] S.J. Kaplan, Cooperative Responses From a Portable Natural Language Database Query System, in *Computational Models of Discourse*, eds. M. Brady and R.C. Berwick, (MIT Press, 1983), pp.167-208.

[7] Johanna D. Moore and William R. Swartout, A reactive approach to explanation, in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, (1989), pp.1504-1510.

[8] C.L. Paris, Tailoring Object Descriptions to a User's Level of Expertise, in *Computational Linguistics*, **Vol.14, No.3** (1988), pp.64-78.

[9] ICOT, Conversation Management System based on Contextual Understanding, in *ICOT Annual Report*, (1988).

[10] H. Yasukawa, H. Suzuki and N. Noguchi, Knowledge Representation Language based on Situation Theory, in *Programming of Future Generation Computers II*, eds. K. Fuchi and L. Kott, (Elsevier Science Publishers B.B., North-Holland,1988), pp.431-460.