

# Repairing Conceptual Mismatches in Human–Computer Dialogue

Robbert-Jan Beun and Rogier M. van Eijk  
*Institute of Information and Computing Sciences*  
*Utrecht University*

A computational framework is presented for the generation of elementary speech acts to establish conceptual alignment between a computer system and its user. This article clearly distinguishes between 2 phases of the alignment process: message interpretation and message generation. In the interpretation phase, presuppositions are extracted from the user's message and compared with the system's semantic model of the application domain, which is represented in Type Theory. Subsequently, in the generation phase, a feedback message at the conceptual level is produced to resolve detected discrepancies and avoid potential discrepancies as a result of quantity implicatures. A conversational strategy is provided that is based on Gricean maxims and a differentiation of various types of information states of the system, such as private and common beliefs about the domain of discourse.

Contemporary technological developments of interactive systems and the expansion of bandwidth enable designers to incorporate a variety of media and modalities in the computer interface. However, merely adding amazing technological feats or increasing bandwidth does not necessarily improve the communication process. When we interact with computers, we also want them to be endowed with characteristics that closely mimic human communication. One of these characteristics is the ability of humans to react in a cooperative manner to the communicative actions of the dialogue partner. In everyday conversation, people effortlessly answer questions, accept or deny assertions, confirm the receipt of a message, and provide relevant feedback in case of communication problems. Because the cognitive and communicative abilities of humans are so well adapted to the real-time processing of these various interaction structures, we expect that including natural

---

Correspondence should be addressed to Robbert-Jan Beun, Institute of Information and Computing Sciences, Utrecht University, P.O. Box 80089, NL–3508 TB, Utrecht, The Netherlands. E-mail: [rj@cs.uu.nl](mailto:rj@cs.uu.nl)

conversational skills in interfaces may contribute to a more efficient and satisfactory human–computer interaction.

One of the prerequisites for natural human communication is that participants are able to reason about and to discuss various aspects of the domain of discourse. To cope with the complexity of the world around us, people consider the existence of objects, discuss possible behavior, draw conclusions from the various dialogue contributions, discuss the meaning of the communication symbols, and many more. Behind these manifestations of various beliefs and opinions is the participant's need to conceptualise the problem domain and to build a coherent and consistent mental model to achieve some sort of common understanding of our complex world.

The situation in human–computer interaction hardly differs from the communicative situation in the real world. In a computer domain, a user should know that there are objects like files and folders, that files are removable, readable, editable, storable, and so forth. Although the need to discuss these various aspects of the virtual domain is probably even more compelling than in the real world, these conversational skills are usually absent from the computer interface.

The goal of this article is to present a computational method that enables us to generate feedback utterances that regulate the avoidance and repair of conceptual mismatches between a computer system and its user. We clearly distinguish between two phases of the alignment process: message interpretation and message generation. In the interpretation phase, assumptions are extracted from the user's message on the basis of the system's belief about the application domain and the rules of well-formedness of these beliefs. In other words, based on both its own semantic model of the application domain and the user's utterance, the system supposes that the user makes particular assumptions about the domain. If conceptual mismatches between the user's assumptions and the system's belief are detected, a feedback message is produced to resolve the mismatch. We provide a conversational strategy that is based on Gricean maxims and the system's dynamic mental state that contains information about the application domain and the conversational partner. The semantic model of the system's beliefs of the application domain is represented in Type Theory (TT). In order to regulate the communicative behavior of the system, we also introduce a model that contains pragmatic knowledge with respect to the user, such as the beliefs and goals of the user. In fact, the pragmatic model marks the semantic information as part of particular information states.

The remainder of this article is organized as follows. First, in the following section we consider the characteristics of the various communication models that should play a basic role in human–computer interaction. We introduce the formal semantic model (FSM), which represents a variety of aspects of the domain of discourse and the formal pragmatic model (FPM). Then, the next section discusses the role of the Gricean cooperation principle in the feedback generation process.

The next two sections then describe the formalization of the semantic model in terms of TT and the detection of conceptual mismatches. Following those sections, the actual conversational strategy is presented. The final two sections wrap up with a discussion and directions for future research.

## MENTAL, CONCEPTUAL, AND FORMAL MODELS

Humans carry a model of the external reality and are able to reason about various aspects of the world, to think about the past and the future, and they can decide which action is appropriate given a certain goal within the circumstances of the activities. In the same manner, users of a particular computer application build their own conceptualization of the characteristics and the behavior of the application. The mental representations that reflect the user's understanding of a system is often termed a mental model. Mental models are used to predict the system behavior and to guide the user's actions.

Mental models are based on the user's previous knowledge and experiences, and evolve naturally with the interaction of the system under consideration. They are sometimes being derived from idiosyncratic interpretations of the system and must operate within the constraints of the human processing system (Norman, 1983). Because only a fraction of the computer application and its internal state is observable, mental models are not only dynamic, they are also often inaccurate, incomplete, inconsistent, and incoherent.

In this article we are not concerned with a detailed analysis of the characteristics of a mental model. In other words, whether a mental model is a picture in the head, a set of propositions, a schema, structural, functional, or any other representation is irrelevant here (cf. van der Veer & Puerta Melguizo, 2002). What is important, however, is that we assume a close relation between the intentional behavior of users, in particular their communicative actions and their mental model. For instance, if a user utters the question, "Restart the system," he or she believes, among many other things, that the system exists and that it can be restarted; if a user clicks on the "save" button after editing a file, we assume that the user wants to apply the action "save" to the file object unless we have evidence to the contrary. Note that the verbs "believe" and "want" are natural language terms that explicitly refer to the internal state of the user. Unintentional non-verbal behavior may as well reveal particular aspects of a user's mental state—think of the application of lie detectors or particular computer games—but valid conclusions are probably much harder to draw from this type of information. Although we expect that most of our results can be generalized into a theory of action sequences in direct manipulation, we concentrate on the intentional communicative behavior of users in terms of simple sequences of words as part of a linguistic dialogue.

In contrast to the concept of a mental model is Norman's (1983) idea of a conceptual model. The conceptual model characterises the relevant objects, their features, relations, behaviors, and the interactions with the user. It is devised as a tool for the understanding or teaching of the system to the user and often informally communicated in natural language, graphical symbols, and pictures. It is important to note, however, that in this article we assume that the conceptual model is usually "in the head" of the designer and, therefore, not directly accessible for other participants, although it looks as if the communication symbols are the conceptual model. In other words, in our view the conceptual model is not a description; it is not on paper, not in the interface, or in any other symbolic form; the symbols and their interrelations are only a means to convey the model to the user.

Designers also are subject to inconsistencies and inaccuracy with respect to their own design, especially in case of complex applications where a team of designers is involved. The role of the conceptual model is, therefore, pragmatically defined; we accept a particular model as the conceptual model and hope that, with respect to a particular task, this model closely resembles the reality in terms of behavior and characteristics of the application.

In our approach, we advocate the idea that we need an explicit representation of the conceptual model inside the machine (for similar approaches, e.g., see Ahn, Beun, Borghuis, Bunt, & van Overveld, 1995; Rich & Sidner, 1998)—that is, a formal counterpart of the conceptual model that is accepted as the expert model; we refer to this model as the formal semantic model (FSM). The FSM contains semantic knowledge with respect to the domain of discourse, for instance, ontological information that helps to catalogue and distinguish various types of objects in the domain, their properties, relations (Sowa, 2000), and assertional knowledge in terms of the concepts defined in the ontology.<sup>1</sup> The FSM, which is modeled in the following as a type theoretical context, is assumed to be semantically grounded in the agent's ability to recognize the inhabitants of certain types in the external world. We distinguish five types of semantic information:

- Information about the existence of a particular type (e.g., "animals exist").
- Information about subtypes (e.g., "mammals are animals").
- Information about the existence of predicates that are applicable to a particular type (e.g., "warm-bloodedness is applicable to animals").

---

<sup>1</sup>A knowledge designer may choose from different formalisms to represent the formal semantic model. In semantic Web applications, for instance, a popular formalism is description logic. In description logic, a so-called TBox contains a set of terminological axioms, and an ABox contains the assertional knowledge (Baader, McGuinness, & Patel-Schneider, 2003). In the type theoretical formalism used in this article, the distinction between TBox and ABox is determined by the type assignment of the expressions.

- Information about rules that state that an object of a particular type has a necessary feature (e.g., “all mammals are warm-blooded”).
- Information about instances of the previous information (e.g., “flipper is a dolphin”).

To regulate the communicative behavior of the system, we also introduce a so-called formal pragmatic model (FPM). The FPM contains pragmatic knowledge with respect to the user, such as the words that can be used during the dialogue and the beliefs and goals of the user. In fact, the FPM marks the semantic information as part of particular mental states; we distinguish five types of pragmatic information:

- Common vocabulary ( $\Gamma_V$ ): the words that can be used in the communication.
- Private belief ( $\Gamma_B$ ): the system’s subjective beliefs about the domain of discourse.
- Common belief ( $\Gamma_C$ ): the part of the system’s belief assumed to be shared with the user.
- Pending belief ( $\Gamma_P$ ): the system’s belief about the user’s belief.
- Goal ( $\Gamma_G$ ): the system’s belief about the user’s goal(s).

To communicate, there needs to be a certain degree of compatibility between the system and the user; this is expressed by the common vocabulary and the common beliefs (e.g., see Ahn, 2001). The common vocabulary contains the terminology of common concepts and enables the system and user to communicate simple messages.

In line with Clark and Schaefer (1989), Allwood, Nivre, and Ahlsen (1992), and Traum (1994), we assume that successful communication requires some degree of conceptual alignment or common ground and that the goal of grounding of the semantic information is a vital activity in cooperative communication. Prior to a conversation, participants not only have beliefs of a particular discourse domain, they also assume that there is some agreement about these beliefs and they augment manifested beliefs to the “agreed beliefs” during the conversation—these agreed beliefs are called the common belief. In practice it is hard (and, because we have no direct access to our dialogue partners, even impossible) to decide whether common beliefs are really common, but our main point is that dialogue participants act as if these beliefs are common. Among other things, the distinction between private and common beliefs enables us to give concrete form to the Gricean maxim of quantity. If relevant, private beliefs can always be manifested unless they are part of the common beliefs. Common beliefs give us a criterion to leave out particular information in the dialogue move (otherwise, we would manifest information that the user already believes).

It is important to note that the information from the FSM cannot be distributed in a random manner over the belief states of the system. For instance, if the system has a common belief that a predicate is applicable to a particular type, the existence of the type should first be introduced in the private beliefs. We assume that the system's information state is organized in a well-formed manner (i.e. according to the rules of the type system; see also Borghuis, 1994). Also, the words in the common vocabulary correspond with variables (e.g., ideas or concepts) declared in the common beliefs of the system.

System beliefs about the user's beliefs will be represented by a temporary belief state, called pending belief, that exists during the process of interpreting the user's message. Pending belief can be considered as a temporary state to separate the user's beliefs from the system's belief. If the information is accepted, it will be transferred to the system's common belief after the system responds to the user's utterance. If the user's assumptions contradict the FSM, they will be kept separate from the system's belief state and rejected if no agreement can be found.

The system's belief about the user's goal gives relevance to the communication process and may be considered as the driving force behind the detection process of the potential mismatches.

Before we explain the details of the semantic model in TT, the detection mechanism of mismatches and the conversational strategy in terms of the Gricean maxims, we first briefly elaborate on the feedback process in dialogue.

## FEEDBACK GENERATION IN DIALOGUE

The term *feedback* originates from the area of cybernetics and refers to the information that a system receives from its environment about the consequences of its behavior. Feedback information is often used to regulate the behavior and guides, in case of purposeful behavior, the actions toward a particular goal. In communication, feedback is used for a broad range of responses at various levels and has an enormous diversity, varying from a simple nod in human-human communication, a particular bit that indicates the receipt of a message in computer-computer communication, to a written comment that evaluates the quality of a scientific article. However, for various reasons, we have no accurate mathematical theory for adequate communicative behavior and the application of cybernetic models to communicative activities has only a limited scope of relevance (Spink & Saracevic, 1998).

When we look at general feedback phenomena in conversations between humans, sequences in terms of speech acts appear to be rather chaotic and seem hardly subjected to any rules. Questions can be followed by answers, denials of the relevance of the question, rejections of the presuppositions of the question, statements of ignorance, and so on (e.g., see Levinson, 1983). An example of general

principles underlying cooperative contributions in dialogue are the Gricean maxims (Grice, 1975) for conversation such as “tell the truth” (quality), “say enough, but not too much” (quantity), “be relevant” (relevance), and “use the appropriate form” (manner). Also, in human-system interaction—where a system is represented by some kind of electronic equipment, such as a computer or a video player—a diversity of heuristics for feedback is suggested. Nielsen (1993), for instance, stated that a system should continuously inform the user about what it is doing and how it is interpreting the user’s input. More detailed heuristics concern the different degrees of persistence in the interface, response times, and corrective feedback in case of errors.

Focusing on the linguistic aspects of feedback in human-system interaction, several authors have modified and extended the Gricean maxims and Nielsen’s (1993) heuristics at various levels of the communication process. Ainsworth and Pratt (1992), for instance, discussed two feedback strategies at the perceptual level for error correction in speech recognition systems in noisy environments. Based on a classification derived from Allwood et al. (1992) and Allwood (1995), Larsson (2003) presented a human-computer feedback system where the levels “contact,” “perception,” “understanding,” and “reaction” are specified. Bernsen, Dybkjær, and Dybkjær (1996) presented an extension of the Gricean maxims based on a corpus of transcribed dialogues obtained from Wizard of Oz experiments. They supplement the maxims with several rules such as “partner asymmetry” (e.g., only 1 partner has a hearing deficiency), “background knowledge,” and “repair and clarification.”

More formal and system-oriented approaches in natural language feedback can be found in, for instance, Hirst, McRoy, Heeman, Edmonds, and Horton (1994), Gabsdil (2003), Schlangen (2004), and Purver (2006) who focused on various aspects of the clarification process of uncertainties and misunderstandings in dialogue. A corpus study of clarification requests by Rieser and Moore (2005) showed that both form and process of the clarification depends on the type of dialogue, such as everyday conversation or task-oriented dialogue. Other researchers, such as Walker et al. (2004), stressed the use of semantic and pragmatic models in terms of strengths of the evidence and user preferences to tailor responses to the conversational partner. Walker et al. showed that user tailored generation of evaluative responses based on multi-attribute decision theory significantly improved the system’s responses in a multimodal restaurant selection task.

Just as the Gricean maxims and their extensions may form guidelines for the acceptability of conversational sequences in human-computer dialogue, Nielsen’s (1993) heuristics offer a practical handle for a systematic evaluation of user interfaces. However, both types of rules are underspecified in case an interface designer wants to realize the actual implementation. In other words, the rules have some explanatory power but no predictive power; they do not provide the designer with

U: “Restart the Internet, please.”

S1: “Sorry, I don’t understand you.”

S2: “What do you mean by ‘restart’?”

S3: “‘Restart’ is only applicable to processes.”

S4: “The Internet cannot be restarted.”

S5: “The Internet is not a process.”

S6: “Networks are not processes.”

S7: “No, because I cannot restart the Internet.”

S8: “Do you believe that ‘restarting’ is applicable to the Internet?”

S9: ...

FIGURE 1 The presuppositions of the user’s questions in conflict with the system’s conceptualization. Note. Depending on its beliefs, the system has abundant possibilities to respond. U = user; S = system.

sufficient detail about the type, content, and form of the feedback that has to be generated in a particular situation.<sup>2</sup> Suppose, for instance, that user *U* and computer system *S* have two disparate conceptualisations and that *U* makes the following request: “Restart the Internet, please.”<sup>3</sup> The system’s FSM contains, among other things, a representation for the words *Internet* and *restart*, and knows that Internet is a subclass of networks. *S* also knows that restarting is only applicable to processes and that networks are not processes. Assuming that our computer system should react in a relevant and truthful way, then what should the response of *S* be? Clearly, we have abundant possibilities for feedback (see Figure 1).

Which utterance is the most adequate one and which rules should be applied to generate these feedback sequences depends, among other things, on the FPM (e.g., on what the user and system know about the application domain and, more specifi-

<sup>2</sup>In fact, Grice (1975) never intended to use the maxims for the *generation* of cooperative sequences in human dialogue, but identified them to show that *violation* of the maxims *generates* inferences that go beyond the semantic content of the utterance. We believe that sophisticated dialogue systems in the future may do the same, but only when these systems are “aware” of the basic principles of dialogues. In our case, the dialogue rules will prevent the user from unwanted inferences.

<sup>3</sup>The example was taken from [www.computerflaters.nl](http://www.computerflaters.nl). A client was asking the helpdesk the following: “Het Internet is erg langzaam. Kunnen jullie het niet opnieuw opstarten?” (“The Internet is very slow. Can’t you restart it?”).

cally, on the system's knowledge about the user's conceptualization). For instance, in S6 (Figure 1) the response is inadequate if the system does not believe that the user's mental model does not contain the information that the Internet is a network. Another parameter is the role played by the system in the interaction; usually, the system acts as an expert who is unwilling to adjust its own FSM, in S8; however, the system reacts as an equal who seems to be willing to reconsider its domain ontology. This seems to be in line with the findings by, for instance, Rieser and Moore (2005) and Walker et al. (2004) that adequate feedback heavily depends on contextual information such as the dialogue situation and the participants' beliefs, goals, and preferences.

Feedback about conceptual mismatches has been studied by several authors. Joshi (1983), for instance, reviewed different types of cooperative responses in question-answer systems and distinguished between responses with regards to so-called *extensional* and *intensional* disparities between the views of the user (U) and the system (S). Extensional disparities are related to the content of the database (e.g., false assumptions by the user), whereas intensional disparities are related to the structure of the database (e.g., assumed non-existing relations by the user). In the end of his article, Joshi generalized his findings into the following modification of the Gricean quality maxim (in the following, MB is an abbreviation for the common beliefs of system and user):

... it should not be possible for U, from what S has said, say Q, to infer something which S believes to be false. If there is such a possibility then after saying Q, S should provide further information to "square away" the MB's. (p. 238)

In other words, the system should avoid inferences by U that are inconsistent with the system's beliefs by choosing a more appropriate response or by adding extra information. For instance, the system's rejection S7 in Figure 1 may imply that the Internet can be restarted in general, but that this particular system is unable to do so. Therefore, to avoid the unwanted inference, a more appropriate response would be S4 ("The Internet cannot be restarted"). Although Joshi's article does not contain any details with respect to the generation process of specific responses, the modification of the Gricean rule will play an important role in our article.

More details of the generation process have been worked out by, for instance, McCoy (1988, 1989), where two types of misconceptions were considered: misclassification of objects (e.g., U: "I thought whales were fish") and misattributes (e.g., U: "I thought whales had gills"). In both articles, a number of response strategies were abstracted from a transcript study and associated with a structural configuration of the user model. McCoy's (1988, 1989) study of the transcripts revealed that a response to a misconception can be viewed as consisting of three parts: (a) a denial of the incorrect information, (b) a statement of the correct

information, and (c) a justification for the denial and the correct response given. To formulate the appropriate response, the proposed generation process heavily relied on the notions of “perspective” (i.e., context sensitive information, such as the goal of the discourse) and “object similarity” (i.e., a similarity metric based on the common and disjoint features of objects involved). Contextual information seems indeed indispensable to an adequate correction process, but the measurement of perspectives and object similarity is rather *ad hoc* and can hardly be verified in different situations because of a lack of details. Another shortcoming is that McCoy’s (1988, 1989) approach does not include an explicit representation of the user’s mental model in terms of various types of beliefs.

### THE FORMAL SEMANTIC MODEL IN TYPE THEORY

In this article, the formal semantic model is expressed in Type Theory (TT). TT, which is actually based on typed  $\lambda$ -calculus, is a powerful logical formalism in the field of theorem proving and programming languages; there is a growing interest in using TT as a framework for natural language semantics (e.g., see Ginzburg, 2005a, 2005b; Ranta, 1994). In Ahn (2001), for instance, it has been shown that *discourse representation structures* (Kamp, 1981) and their classical semantics can be seen as a limit case of a more general theory of types. In the field of agent communication, TT was used in the DenK project as a knowledge representation to model various types of beliefs (Ahn et al., 1995; Bunt, Ahn, Beun, Borghuis, & van Overveld, 1998). In the project, an “intelligent” agent was modeled that supported a human user in its use of a particular domain. Although the system was applied to the domain of an electron microscope, it was intended to be generic in that its architecture and various techniques for knowledge representation and construction were independent of the field of application. The agent’s belief states, such as private and common beliefs, were modeled as type theoretical contexts. In the DenK project, the formalism was used to model the ontological assumptions and beliefs about the task domain (the electron microscope) and the cognitive dynamics of the agent’s belief state; in particular, the change of beliefs as a result of domain observations and dialogue contributions by the user.

Apart from its intrinsic dynamic properties, TT has important advantages over formalisms such as predicate logic or discourse representation theory. First, and at the heart of TT, is the formal distinction between objects and types. Types often represent a particular concept, whereas objects can be considered as instances of these concepts. In TT it can be expressed, for instance, that we have the concept “network” and that “the Internet” is an instance of the concept network. Second, TT embodies the notion of contextuality (or sequentiality) that tells us that new assumptions can only be added as long as the current belief state satisfies particular constraints.

To some extent, the notion of sequentiality corresponds to the notion of *presupposition* used in linguistics (Strawson, 1950) where, for instance, the sentence, “Type Theory will be the most prominent logical framework in the near future to model the semantics of natural language utterances” presupposes “the existence of logical frameworks,” “type theory as a particular instance of these logical frameworks,” “the existence of natural language utterances and their semantics,” and so on. Presuppositions can be considered as a kind of background assumptions that should be fulfilled to understand the meaning of the message (see also Piwek & Krahmer, 2000). An important characteristic of presuppositions is that they can be inferred from both the sentence and its interrogative form. Because we assume a close relation between the communicative actions of the users and their mental model, presuppositions give us valuable information about the discrepancies between the FSM and the user’s model of the application. Hence, we assume that the system is able to interpret a simple language fragment with words taken from the common vocabulary and that it detects mismatches on the basis of presuppositions derived from the message.

### Semantic Information

We consider how the five types of semantic information can be expressed in a very limited fragment of TT (e.g., for comprehensive introductions, see Ahn, 2001; Luo, 1994). The building blocks in TT are expressions, called *statements*, of the following form:

$$G: H$$

that indicate that object  $G$  is an inhabitant or instance of type  $H$ , or that  $G$  has type  $H$ , for short.

How can we express the first type of semantic information: “the existence of a particular type?” For this there is a special type named *sort* (which is denoted by  $*s$ ). To say that a particular type exists is to say that it is an inhabitant of this special type sort. For instance, the statement:

$$\textit{animal}: *s$$

expresses that animal is a type; in other words, that the type animal exists.

The type  $*s$  itself is an inhabitant of the top-level type “ $\square$ .” This is expressed by the following statement:

$$*s: \square$$

To express the information about the existence of predicates that are applicable to a particular type, another special type is introduced: the sort of propositions, which is denoted by  $*p$  (which itself is an inhabitant of the top-level type  $\square$ —denoted by  $*p: \square$ ). For instance,

$$\text{warm\_blooded}(\text{flipper}): *p$$

expresses that *warm\_blooded(flipper)* is a “proposition.” It does not yet state that it is a true proposition. In TT, propositions are considered as types themselves, and a proposition is true if there exists an object (i.e., the proof) with the proposition as its type. For instance, the statement:

$$x: \text{warm\_blooded}(\text{flipper})$$

expresses that the proposition *warm\_blooded(flipper)* is true, where *x* denotes the particular proof object.

Predicates are modeled as functions that take inhabitants of a particular type as their argument and that yield propositions as their result. For instance, the following:

$$\text{animal} \rightarrow *p: \square$$

indicates the existence of predicates over the type of animals.

For the third type of semantic information, “information about subtypes,” a subsumption operator, “<” is introduced, which indicates that inhabitants of a more specific type can be applied in every case where inhabitants of the more general type may be applied (types on the left are lower in the hierarchy):

$$\text{mammal} < \text{animal}: *s$$

$$\text{dolphin} < \text{mammal}: *s.$$

Therefore, any predicate that takes as an argument inhabitants of the type *animal* can take inhabitants of the type *mammal* as well. Similarly, any predicate that takes as an argument inhabitants of *mammal* can take inhabitants of *dolphin* as well.

Information about rules that state that an object of a particular type has a necessary feature is modeled by means of the product operator “ $\pi$ .” In general,  $\pi x:A.P(x)$  is the type of rules that yield that all instances of the type *A* necessarily have the feature *P*.<sup>4</sup> Therefore, for instance,

$$\pi x:\text{mammal}.\text{warm\_blooded}(x) : \square$$

expresses the type of rules that yield that all mammals are warm-blooded.

Finally, we consider the fifth type of semantic information: information about instances of these various types of information. The information that *flipper* is a *dolphin* is modeled as *flipper* being an object of the type *dolphin*:

---

<sup>4</sup>In fact, the types  $\pi x:A.P(x)$  and  $A \rightarrow *p$  are both instantiations of the more general product type  $\pi x:AB$ . However, for the purposes of this article we do not need to consider this extra level of abstraction. For more details see Ahn (2001) and Kievit (1998).

*flipper: dolphin.*

The information that warm-bloodedness is a predicate that is applicable to the type of animals is expressed by the following:

$$\text{warm\_blooded: animal} \rightarrow *p.$$

The information that all mammals are warm-blooded is expressed by the statement:

$$\text{zoological\_rule}_{73}: \pi x:\text{mammal}.\text{warm\_blooded}(x)$$

indicating that *zoological\_rule*<sub>73</sub> is a function that for each instance *x* of the type mammal yields a proof object for the proposition *warm\_blooded*(*x*).

## The Type System

The various types of beliefs of an agent can be represented in TT as so-called contexts; contexts consist of sequences of expressions and list everything that has been assumed so far by the system with respect to the application domain and everything that has explicitly been inferred from these assumptions. The notion of sequentiality plays a role in the order in which statements can be added to contexts. For instance, a statement like *flipper: dolphin* cannot be added unless the context already contains the statement *dolphin: \*s*.

Contexts are thus open to new information and can be extended as long as new introductions are adhered to the rules of the type system; these contexts are called *legal contexts*. Therefore, given a particular context, the rules of the type system constrain the way in which statements can be combined into new legal statements. This can be expressed by so-called judgments:

$$\Gamma \vdash G: H$$

that expresses that object *G* has type *H*, given the assumptions in context  $\Gamma$ . The context  $\Gamma$  is called the *assumption* of the judgment, and the statement *G: H* is called the *conclusion*.

A context  $\Gamma$  is defined to be legal if

$$\Gamma \vdash *s: \square.$$

Because the statement *\*s: □* is an axiom of the type system, this judgement boils down to the requirement that the context has been constructed according to the rules of the type system. In addition, we call a statement *X* a *legal extension* of a context  $\Gamma$  if the extended context is legal; that is,

$$\Gamma, X \vdash *s: \square.$$

TABLE 1  
 Derivation of the Legal Context  $\Gamma$  Containing  $zr_{73}$  (*flipper*):  
*warm\_blooded(flipper)*

No.	Context	Derivation
1.	[*s: $\square$ ,	(axiom)
2.	<i>animal</i> : *s,	(1.), (start1)
3.	<i>animal</i> $\rightarrow$ *p: $\square$ ,	(2.), (pred)
4.	<i>warm_blooded</i> : <i>animal</i> $\rightarrow$ *p,	(3.), (start1)
5.	<i>mammal</i> : *s,	(1.), (start1)
6.	<i>dolphin</i> : *s,	(1.), (start1)
7.	<i>flipper</i> : <i>dolphin</i> ,	(6.), (start1)
8.	<i>mammal</i> < <i>animal</i> : *s,	(2.), (5.), (start2)
9.	<i>dolphin</i> < <i>mammal</i> : *s,	(2.), (6.), (start2)
10.	$\pi x$ : <i>mammal.warm_blooded</i> (x) : $\square$ ,	(4.), (start1), (8.) (apply1), (prod)
11.	$zr_{73}$ : $\pi x$ : <i>mammal.warm_blooded</i> (x),	(10.), (start1)
12.	$zr_{73}$ ( <i>flipper</i> ): <i>warm_blooded</i> ( <i>flipper</i> )]	(11.), (7.), (9.), (apply2)

Judgments are derived from the rules of the types system. In general, a derivation rule is of the following form:

$$J_1 \ \&\dots\ \& \ J_n \ \Rightarrow \ J$$

expressing that the judgment  $J$  (called the *consequent*) is derivable if each of the judgments  $J_1 \dots J_n$  (called the *antecedents*) is derivable. The formal derivation rules of the simple type system that we use in this article can be found in the Appendix.

We illustrate the rules of the type system and the notion of legality by means of an example. Table 1 depicts the construction of a legal context  $\Gamma$  that contains the proof of the proposition that flipper is warm-blooded; that is, the last statement (12) indicates that the application of zoological rule number 73 to the object *flipper* constitutes the particular proof object. To be able to derive this statement, according to rule (apply2) of the type system, this zoological rule has to be of the appropriate product type. The zoological rule takes instances of the type *mammal* (11), and this is okay because *flipper* is of type *dolphin* (7), which is a subtype of *mammal* (9). According to rule (start1), it is required that this product type exists (10). For this it is required according to rule (prod) in combination with rules (apply1) and (start1) that the predicate *warm\_blooded* is applicable to the type *animal* (4) and that *mammal* is a subtype of *animal* (8). For the subtype relations (8) and (9), it is required according to rule (start2) that the types *animal* (2), *mammal* (5), and *dolphin* (6) themselves exist. Likewise, for the predicate *warm\_blooded*, it is required according to rule (start1) that the type of predicates over animals exists (3), for which in turn according to rule (pred) the type *animal* should exist (2). Finally,

(2), (5), and (6) are using rule (start1) derivable from the axiom (1). As the construction of this sequence of statements (1) through (12) adheres to the rules of the type system, it constitutes a legal context.

## CONCEPTUAL MISMATCHES

In the literature, various typologies of conceptual mismatches have been identified (for instance, see Agarwal, Huang, & Dimitrova, 2005; Hameed, Sleeman, & Preece, 2002; Visser, Jones, Bench-Capon, & Shave, 1998). In this article, we distinguish two basic categories of mismatches: *direct* and *indirect* mismatches.

Direct mismatches result from an incorrect match between the private belief of the system's FSM and the presuppositions derived from the user's message. For example, note the imperative:

User: "Restart the Internet, please!"

The user presupposes that the action *restart* is applicable to an object called *Internet*, which is in conflict with the system's FSM. Direct mismatches are a result of the *a priori* conceptualization of the domain of discourse by user and system; the assumptions that cause the mismatch are, therefore, part of the semantic content of the linguistic realization.

To provide feedback about direct conceptual mismatches we need a computational decision criterion that tells us whether information from the user is incompatible with the FSM of the system (Beun, van Eijk, & Prüst, 2004). To achieve this we use the concept of *legality* introduced in the previous section to compare the semantic representation of the incoming language fragment with the FSM of the system, in particular its private beliefs.

Let  $Q$  denote the semantic representation of the user's input question and let  $\Gamma_B$  denote the system's private beliefs about the domain of discourse (which we assume to be a legal context). We define  $Q$  to be *free of direct conceptual mismatches* if the following judgment holds:

$$\Gamma_B, x:Q \vdash *s: \square,$$

where  $x$  is fresh. In other words,  $Q$  is free of conceptual mismatches if it can be instantiated by a fresh object  $x$  such that the resulting statement " $x:Q$ " is a legal extension of the private beliefs  $\Gamma_B$ .

The mechanism for the detection of conceptual mismatches is as follows. First, we assume a function "CONSTRUCT-DERIVATION-TREE" that takes a judgment as its input and computes whether this judgement can be derived from the type system. This function builds a derivation tree by recursively applying the appropriate derivation rules in opposite direction (i.e., from consequent to antecedent).

ents). If all these recursive applications can ultimately be resolved to the axiom of the type system (i.e., all leaves are of the form  $\epsilon \vdash *s: \square$ ), then the tree is said to be *complete*. If, however, one or more judgments cannot be resolved to the axiom, the tree is said to be *incomplete*.

We use this function to detect conceptual mismatches by constructing the proof tree of the judgement  $\Gamma_B, x:Q \vdash *s: \square$ . Indeed, if this yields a complete derivation tree, this implies that  $Q$  is free of conceptual mismatches. However, if the derivation tree is incomplete, then each of the leaves of the tree that are is of the form  $\epsilon \vdash *s: \square$  indicates a conceptual mismatch. Given an incomplete derivation tree, we annotate each of its nodes with a mark “MISM” or “OK,” expressing whether the corresponding judgment indicates a conceptual mismatch. Resolved leaves have the annotation OK; leaves that cannot be resolved have the annotation MISM. In addition, if a node has a node marked MISM as one of its children, then this node has the mark MISM itself and OK otherwise. Finally, the conclusions of the judgments of all nodes marked MISM together denote the conceptual mismatches between the user input and the system’s FSM (except for the root node because it has  $*s: \square$  as its conclusion). These conceptual mismatches become part of the pending beliefs  $\Gamma_p$ .

Therefore, in summary, the procedure for the detection of conceptual mismatches is as follows:

#### COMPUTE-MISMATCHES ( $Q$ ).

1. CONSTRUCT-DERIVATION-TREE ( $\Gamma_B, x:Q \vdash *s: \square$ ).
2. IF the tree is complete.
3. THEN output:= *no mismatches*.
4. ELSE annotate all nodes with their corresponding mark .
5. AND output:= annotated tree.

To illustrate this detection mechanism, we consider the user command “*restart (the\_internet)*.” Suppose the system’s private beliefs  $\Gamma_B$  are as follows:

$$\begin{aligned} \Gamma_B = [ & process: *s, \\ & restart: process \rightarrow *p \\ & network: *s, \\ & internet: *s, \\ & internet < network : *s, \\ & the\_internet: internet]. \end{aligned}$$

For the detection of possible conceptual mismatches we start with the following:

$$\Gamma_B, x: restart (the\_internet) \vdash *s: \square.$$

According to rule (weakening1), we have to derive:

$$\Gamma_B \vdash \text{restart}(\text{the\_internet}): *p,$$

which among others according to rule (apply1) requires the derivation of the following:

$$\Gamma_B \vdash \text{internet} < \text{process}: *s,$$

and this requires, according to successive applications of rule (weakening1), the derivation of the following:

$$\epsilon \vdash \text{internet} < \text{process} : *s,$$

which, however, cannot be further resolved. Thus, the conclusion of this judgment, *internet < process : \*s* constitutes a conceptual mismatch (marked as MISM). Because it is the conclusion of a parent node, the statement, *restart(the\_internet): \*p*, constitutes a conceptual mismatch as well. The full result is summarized in Table 2.

In other words, the user command to restart the Internet presupposes, according to the system’s FSM, that *restart(the\_internet)* is a well-formed proposition (i.e., is of type *\*p*), which, because this predicate is applicable to *process* and *the\_internet* is of type *internet*, in turn presupposes that *internet* is a subtype of *process*, which, however, is not the case, yielding a conceptual mismatch between the user and the system’s FSM.

Although direct mismatches can be automatically detected from the semantic content of the user’s message, indirect mismatches result from the system’s answer. They are caused by pragmatic inferences that can be deduced on the basis of the rules of the dialogue game, such as Gricean implicatures. For instance, imagine a situation where a user observes a number of bottles with the description “toxin.” Suppose that, for whatever reason, the user asks the question:

User: “Is this toxin poisonous?”

TABLE 2  
Computation of Conceptual Mismatches for the User Input,  
“restart(the\_internet)”

No.	Statements	Presuppositions	Annotation	Formal Pragmatic Model
	<i>x: restart(the_internet)</i>			Goal
1.	<i>*s: □</i>		ok	Common vocabulary
2.	<i>process: *s</i>	(2.)	ok	
3.	<i>process: → *p: •</i>	(3.)	ok	
4.	<i>restart: process → *p</i>	(4.)	ok	
5.	<i>internet: *s</i>	(1.)	ok	
6.	<i>the_internet: internet</i>	(5.)	ok	
7.	<i>internet &lt; process: *s</i>	(2.), (5.)	MISM	Pending belief
8.	<i>restart(the_internet): *p</i>	(4.), (5.), (7.)	MISM	

A simple affirmation by the system (e.g., Yes) triggers an inference by the user that may cause a serious conceptual mismatch; namely, that not all toxins are poisonous. The inference can be concluded from the first part of the Gricean maxim of quantity that states that dialogue participants should contribute as much as possible given the goal of the interaction. In fact, the inference is a so-called quantity implicature (Levinson, 1983). If a speaker can contribute a stronger proposition—in this case that toxins are always poisonous—and the stronger proposition also satisfies the other Gricean maxims (i.e., quality, the second part of quantity, relevance, and manner), then the speaker should add, in line with Joshi’s (1983) proposal, extra information. Consequently, if the speaker withholds the stronger information, the hearer may conclude that the information does not hold, especially in cases where the speaker is supposed to be the expert of the discourse. Therefore, to avoid the discrepancy, the system has to add extra information to the affirmation, for example:

System: “Yes, because toxins are always poisonous.”

## A CONVERSATIONAL STRATEGY

In this section, we design a conversational strategy to respond to the user’s input, which detects direct mismatches and avoids indirect mismatches. We start with the processing of the user input. Again let  $Q$  denote the semantic representation of the user’s input and let  $\Gamma_B$  denote the system’s private beliefs. We define the following function:

### PROCESS-INPUT( $Q$ )

1. IF COMPUTE-MISMATCHES ( $Q$ ) = *no mismatches*.
2. THEN find an  $x$  such that CONSTRUCT-DERIVATION-TREE ( $\Gamma_B \vdash x: Q$ ) is complete
  - IF such an  $x$  exists
  - THEN output:= tree
  - ELSE output:= *cannot be proved*.
3. ELSE output:= COMPUTE-MISMATCHES ( $Q$ ).

The function PROCESS-INPUT first checks the presence of direct conceptual mismatches between the user input and the system’s FSM. If a mismatch exists, then it yields the annotated derivation tree. If no mismatches have been found, then a proof object  $x$  for  $Q$  is computed (together with the corresponding derivation tree). If such an  $x$  does not exist, then the function yields the result *cannot be proved*; otherwise, it yields the corresponding derivation tree. Note the subtle but essential difference between the judgments  $\Gamma_B, x:Q \vdash *s: \square$  (with  $x$  fresh) and  $\Gamma_B \vdash x:Q$ ; the former indicates that the statement  $x:Q$  is a legal extension (and thus that

$Q$  is well-formed), and the latter expresses that the statement can be proved (and thus that  $Q$  is true).

Assuming the system to be an expert on the domain, what should its response be? This depends on various aspects, among others the system's *attitude*. In this article, we discern two different attitudes, which are called *lazy* and *eager*. With respect to direct mismatches, a lazy system will report the most prominent conceptual discrepancy (i.e., from the highest node marked MISM in the annotated derivation tree), whereas an eager system will report all conceptual discrepancies (i.e., all nodes marked MISM). When there are no conceptual discrepancies and a complete derivation tree has been constructed, a lazy system will report the most prominent reason (i.e., from the highest node in the derivation tree), whereas an eager system will report the entire proof.

An important question is: What are the requirements w.r.t. the extra information (conceptual discrepancies and reasons) that is included in the response. The answer to this question depends on whether this information satisfies the Gricean maxims. For instance, for the user question, "Is this toxin poisonous?", we have:

- Manner: The system believes that the user knows the words *toxins* and *poisonous*.
- Quality: The system believes that toxins are always poisonous.
- Quantity 2: The system believes that the user does not believe that all toxins are poisonous.
- Relevance: The system believes that the information "toxins are always poisonous" is relevant.

The words toxin and poisonous were part of the user's question, so the first maxim holds. Because we have explicitly assumed that the system believes that toxins are always poisonous, the second maxim holds. The same counts for the third maxim: If the user is cooperative, it may be concluded from his or her question that he or she believes that toxins can be poisonous or not poisonous. Therefore, he or she does not believe that toxins are always poisonous (Quantity 2). The relevance maxim cannot be proven; therefore, and we assume that, if one of the participants wants to know whether an object of a particular type has a particular characteristic, then by default it is always relevant that the participant knows that all these type of objects have that characteristic. It is hard to think of a situation where this is not the case and it certainly holds for the toxin case. Therefore, to avoid the conceptual discrepancy, depending on its attitude, the system should add the extra information.

Depending on both the content of the FSM and FPM, even more informative responses may be generated. Let us consider the simple domain of animals again. Suppose that the user asks the following:

User: "Is this dolphin warm-blooded?"

Then an adequate response could be:

System: "Yes, because all mammals are warm-blooded."

Note that in the response the information that dolphins are a subclass of mammals is included as background information and should be, to be cooperative, part of the common belief of the FSM.

Summarizing, we opt for the following conversational strategy:

#### CONVERSATIONAL-STRATEGY

1. PROCESS-INPUT(Q).
2. IF *conceptual mismatches exist*.
3. THEN output:= *No, because ARGUMENT*.
4. ELSE IF *Q can be proved*.
4. THEN output:= *Q because ARGUMENT*.
6. ELSE output:= *No*.
7. WHERE *ARGUMENT* satisfies the following requirements:
  - a. Is according to the system's attitude (i.e., the relevance maxim).
  - b. Is expressed in terms of the common vocabulary (i.e., the manner maxim).
  - c. Is based on the (annotated) derivation tree (i.e., the quality maxim).
  - d. Is not part of the common belief (i.e., the second quantity maxim: do not say too much).
  - e. All its presuppositions are part of the common belief (i.e., the first maxim of quantity: say enough).

We illustrate this strategy with some examples. First, we consider the case of indirect mismatches. Let us consider this user question:

User: "Is this dolphin warm-blooded?"

In Table 3 we have presented the sequence of statements of the FSM that constitute the derivation tree of the proposition *warm-blooded(this\_dolphin)*. In the table we have indicated for each of the statements its most important presuppositions (excluding \*s: □, for instance) and the part of the FPM it belongs to.

According to the previous strategy, and assuming an eager attitude, the answer to the user's question would be:

System: "(9.), because (8.). ["Yes, because all mammals are warm-blooded."].

TABLE 3  
 Derivation Tree and Formal Pragmatic Model (FPM) for the Generation of  
 the Response to the Question, "Is this dolphin warm-blooded?"

No.	Formal Semantic Model	Presuppositions	Formal Pragmatic Model
1.	$x: warm\_blooded(this\_dolphin)$		Goal
2.	$animal: *s$		Common vocabulary
3.	$mammal: *s$		
4.	$dolphin: *s$		
5.	$warm\_blooded: animal \rightarrow *p$	(1.)	
6.	$this\_dolphin: dolphin$	(3.)	
7.	$mammal < animal: *s$	(1.), (2.)	Common belief
8.	$dolphin < mammal: *s$	(2.), (3.)	
9.	$f: \pi x: mammal.warm\_blooded(x)$	(4.), (6.)	Private belief
	$f(this\_dolphin):$	(5.), (7.), (8.)	
	$warm\_blooded(this\_dolphin)$		

Because (8.) and (9.) are part of the private beliefs and not of the common beliefs, their presuppositions are part of the common beliefs and *mammal* and *warm-blooded* are part of the common vocabulary.

Note that if the information (7.) *dolphin < mammal: \*s* would have been part of the private belief instead of the common belief, then the system's response would have been:

System: "(9.), because (8.) and (7.)." ["Yes, because all mammals are warm-blooded and dolphins are mammals."].

Let us now consider the case that the type "mammal" is not part of the common vocabulary but of the private belief. In particular, let us assume that the distribution of the semantic information over the FSM is as depicted in Table 4.

In this situation, the private belief "all mammals are warm-blooded" is not expressible in terms of the common vocabulary. The strongest information that is expressible in the common vocabulary is the information "all dolphins are warm-blooded." Hence, according to the conversational strategy, the system responds with:

System: "(10.), because ((9.) plus (8.) minus (6.))." ["Yes, because all dolphins are warm-blooded."].

Finally, we address the case of direct mismatches. Consider again the user input:

User: "Restart the Internet, please!"

TABLE 4  
 Derivation Tree and Formal Pragmatic Model (FPM) for the Generation of  
 the Response to the Question, "Is this dolphin warm-blooded?," Where the  
 Word *Mammal* is Not Part of the Common Vocabulary

No.	Formal Semantic Model	Presuppositions	Formal Pragmatic Model
1.	$x: \text{warm\_blooded}(\text{this\_dolphin})$ $\text{animal}: *s$		Goal Common vocabulary
2.	$\text{dolphin}: *s$		
3.	$\text{warm\_blooded}: \text{animal} \rightarrow *p$	(1.)	
4.	$\text{this\_dolphin}: \text{dolphin}$	(2.)	
5.	$\text{dolphin} < \text{animal}: *s$	(1.), (2.)	Common belief
6.	$\text{mammal}: *s$		Private belief
7.	$\text{mammal} < \text{animal}: *s$	(1.), (6.)	
8.	$\text{dolphin} < \text{mammal}: *s$	(2.), (6.)	
9.	$f: \pi x: \text{dolphin.warm\_blooded}(x)$	(3.), (8.)	
10.	$f(\text{this\_dolphin}):$ $\text{warm\_blooded}(\text{this\_dolphin})$	(4.), (8.), (9.)	

and the derivation tree and FSM as have been depicted in Table 2. Then, according to the conversational strategy, the different attitudes give rise to the following two responses<sup>5</sup>:

System(lazy): "No, because not(8.)." ["No, because it is impossible to restart the internet."].

System(eager): "No, because not(7.) and thus not(8.)." ["No, because an internet is not a process and thus it is impossible to restart the internet."].

## DISCUSSION

We have presented a computational model that generates formal conceptual structures that can be used in the generation of cooperative natural language responses in human-computer dialogues. In this article, we have focused on language fragments that should be generated in cases of mismatches between the mental model of the user and the system's conceptual model of a particular discourse domain. For that we have formalized the system's conceptual model in TT, and we have distinguished various types of information states to regulate the system's communicative behavior. In this article, we have taken a strong theoretical stance because we

<sup>5</sup>Note that as a result of different type assignments ( $*p$  and  $*s$ , respectively), different terms can be used in the linguistic realization of the responses "is impossible" and "is not," respectively.

believe that a fundamental approach is necessary to model human-computer interaction, especially in the field of natural language processing.

We now discuss some of the strong and weak points of the presented model. Let us, therefore, first consider some more examples that were taken from [www.computerflaters.nl](http://www.computerflaters.nl).<sup>6</sup> The first example shows that conceptual mismatches are often embedded in various discourse contributions:

Helpdesk: "Helpdesk, how can I help you?"

Client: "I have a complaint."

Helpdesk: "What is the problem?"

Client: "I asked you not to program pornography on my Internet. I want you to remove it immediately!"

The last turn by the client contains several conceptual mismatches, such as "the people from the helpdesk program the Internet," "it is the client's Internet" (whatever that means), and "the helpdesk can remove pornography from the Internet." The example shows that, in practice, a sophisticated discourse preprocessor is required to build up the appropriate discourse model to resolve, for instance, the pronominal reference ("it") and the gap from the elliptical construction ("from the Internet") of the last sentence. The meaning of the client's last sentence would then roughly be:

Client: "I want you to remove pornography from the internet."

In our article, we have focused on only one sentence that is disambiguated with respect to anaphorical reference and elliptical constructions, but a future system should include discourse information as well. Clearly, we do not want the following type of system feedback if the missing argument in the verb *remove* can be filled with information from the previous discourse or other sources:

User: I want you to remove iTunes."

System: I cannot remove iTunes, because you always have to remove something from something else."

In principle, our framework can be extended to discourses of several utterances, but only if the correct preprocessing is applied.

---

<sup>6</sup>Because the original sources are untraceable, we do not know how reliable the examples are. It is not hard to imagine that these examples could have appeared in helpdesk conversations, however. Another shortcoming is that none of the examples that we found, except one, contained a continuation by the helpdesk assistant after the client's introduction of the mismatch, so we have to guess for the appropriate response.

Another point of attention is the behavior of presuppositions in imperative and embedded sentences. From the client's last sentence in the previous example, the system should be able to infer that the client believes the conflicting information that the removal of pornography from the Internet is a possible action by the helpdesk. Given that pornography would be part of the system's vocabulary, the statement, "pornography is removed by the helpdesk from the internet," would be added to the goal state of the system and compared with its private belief. From the system's private belief, it would follow that pornography can never be removed from the Internet by the helpdesk, and a lazy response in natural language would be:

System: "Pornography cannot be removed from the internet (by the helpdesk);"

which seems adequate for the time being. Therefore, in this case the semantic content of the postcondition of the imperative, "Remove pornography from the internet," is put in the goal state of the system. It remains for future research how well this strategy works for different types of embedding and different sentence types.

Another example that we found and that included a response from the helpdesk is the following:

Client: "I want to type capital letter 7."

Helpdesk: "Ciphers do not have capital letters, sir."

Here the system would detect that the client erroneously believes that "7" can be typed as a capital letter. If the statement that 7 is of type cipher would be part of the system's common belief—and depending on other subtleties, such as the represented meaning of the verb capitalize—the system would react as follows:

System: "Ciphers cannot be capitalized,"

which largely agrees with the original response of the helpdesk.

Other example utterances from the client that we found were as follows:

"I want to download the internet."

"I want to send an email to 04123-4567."

"I cannot send email. Is the internet filled up?"

"Always when I move the mouse, the screensaver disappears."

"Is the internet open on Sunday as well?"

Given a correct conceptual model of the system with respect to objects like screensavers, telephone numbers, days of the week, and verbs like open, filled up,

and so on, the system is, in principle, able to detect the previous mismatches and to generate lazy or more elaborate responses.

It is interesting to note that the presented system would not only be able to detect the user's mismatch, but would also be capable of avoiding inconsistencies in its own feedback messages, such as the following:<sup>7</sup>

System: "Error 101: No keyboard. Press F1 to continue."

In this case, the system would believe, for instance, that pressed keyboard buttons like F1 can only control a program if the keyboard is connected to the computer.

So far we have seen that, depending on the conceptualization of the discourse domain, the system is able to detect a wide range of conceptual mismatches and that the system is capable to generate conceptual structures that may form the input for a natural language response generator. How cooperative are these responses? To answer this question, we briefly return to the primary assumption underlying the conversational model of the presented system. In line with many other researchers (e.g., Clark, 1996; Joshi, 1983; Traum, 1994), we assumed that "common ground is a sine qua non for everything that we do with others" (Clark, 1996, p. 92) and that we constantly add information to our common ground during the interaction. Common ground is what conversational partners share, but because nature has separated our minds and no one has direct access to the mental state of the other, dialogue participants have to pretend that part of their individual beliefs is common. They have to work hard to coordinate this common ground, every new piece of common ground is built on an old piece, every new piece of information should be consistent with previous information, and for that sometimes incorrect pieces have to be removed. The feedback process that regulates the coordination of the common ground is therefore an inevitable part from every natural language dialogue system. From this it would follow that the repair of mismatches as presented in this article is an essential part of the process that we call cooperative conversation.

Cooperativity is not a Boolean operator but manifests itself as a sliding scale, however. Sometimes extra information is more cooperative, sometimes less; sometimes goal-directed responses are cooperative, sometimes they are not. All this heavily depends contextual aspects, for instance, on the role the system plays in the interaction such as an expert, a teacher, or a helpful friend. Depending on the role, we would expect different conversational strategies: A teacher may be more focused on the creation of the common ground and may, therefore, give more general

---

<sup>7</sup>Imagine in future systems a central dialogue manager that receives and collects messages from applications in a multi-agent setting and that presents itself as a single agent to the user. Beforehand, it would be impossible to guarantee that these different applications are always consistent with respect to their feedback messages.

and elaborate feedback than the expert who focuses only on those parts of the common ground that are related to the task at hand. The presented model enables us to experiment with different types of attitudes such as lazy and eager. A slightly different but related problem is the following. Remember the first example in this section where the client speaks about “my internet.” In this context, the system would be considered a bore if it would respond with, “Internet does not belong to you”; but, in a context of properties and trade, it could be crucial to deny the presupposition.

Also, in contrast with a helpful friend role, we do not expect a system that plays an expert role to change its conceptual model as a result from the interaction with the user. In fact, an important feature of our system is that we implicitly presumed so-called closed world assumption: If particular information cannot be derived from the belief state of the system, it decides that it is not the case. In TT this can be expressed as follows: If restart is not applicable to the Internet, it can never be a proposition; if it can never be a proposition, there will never be a proof object; if there will never be a proof object, it can never be true, therefore, it is false in all possible worlds. This is probably not the behavior that we observe in most human conversations, but for the moment it seems a satisfactory and cooperative strategy in the case of human–computer interaction.

Cooperativity largely depends on the goals of the conversational partners and, therefore, the system should be “aware” of the user’s goals that go beyond the utterance; the absence of a sophisticated goal-handling subsystem is an important drawback in our system. On the other hand, whatever the goals of the dialogue players are, they always have the meta-goal of coordinating the common ground (i.e., to carefully construct a common belief state based on pieces of previous knowledge and to avoid mismatches whenever possible). In some cases it may even be more cooperative not to respond to some assumed goal. For instance, although in some contexts the following system response looks perfectly adequate:

User: “Restart the internet!”

System: “To speed up your system you have to close down the backup program,”

there may be many other reasons why the user wants to restart the Internet; in those cases the response would be completely irrelevant. Therefore, the lazy response, “It is impossible to restart the internet,” is always on the safe side; and, depending on other parameters, the system may decide to give more information. Because the type theoretical framework enables us to keep track of the proofs for every statement, the extra information can be postponed to a next turn, for instance, if the user asks a reason for the system’s response. Therefore, depending on the common ground of the system, the dialogue may continue as follows:

User: “Restart the internet!”

System: "It is impossible to restart the internet."

User: "Why is that?"

System: "Because the internet is a network and networks cannot be restarted."

A more helpful response without explicit goal handling could be generated if the system would be able to infer objects in the common field of attention that semantically fit the restart action and offers them as an alternative to the user, for instance:

User: "Restart the internet!"

System: "Do you want me to restart the internet browser or to restart the computer?"

However, suppose we have 20 of these objects or 1,000, then what would be a cooperative response? One way is to narrow down the number of "fitting" objects by the introduction of some sort of focus space that would contain the objects that were discussed in the dialogue or objects that are part of a common field of perception. An argument against this strategy is that the user correctly uses the object "Internet," but incorrectly uses the verb restart. This would imply that we should also offer the verbs that semantically fit the Internet in the response. In all cases we need extra information, such as stress and focus spaces, to calculate these possibilities. We believe that this is an interesting extension and that it can be added in a second, less rudimentary version of the model, but it is beyond the scope of the current article (cf. McCoy, 1989, or Walker et al., 2004).

## CONCLUSION

From a point of view of human-computer interaction developers, one of the challenges is that systems are designed in such way that they support the user's acquisition of an appropriate mental model to avoid errors while performing on them. In this article, we have presented a model that enables a computer system to generate speech act sequences on a conceptual level in case of mismatches between the system's semantic model of the application domain and the user's mental model. For that we have modeled the system's model in TT, and we have distinguished various types of information states to regulate the system's communicative behavior.

We expected that including natural conversational skills in interfaces may contribute to a more efficient and satisfactory human-computer interaction. Due to its complexity, natural language interaction with computer systems is usually avoided and, if included, highly underdeveloped. In this article, we have taken a strong theoretical stance; however, to determine what humans actually do in realistic conversational circumstances and to validate the model presented in this article, the acquisition of empirical data is a necessary step in the research process. We have,

therefore, planned to collect empirical data from help desk conversations between experts and naive users. We expect that the analysis of the transcripts will lead to a further extension and refinement of the model, such as the inclusion of a discourse model and richer semantic descriptions. To prevent irrelevant linguistic realizations in the generation process, we believe that including various types of information sources, such as the topic of the conversation, the user's goals and preferences, should also be taken into consideration.

Our proposal should be considered as one of the many steps in the design of a cooperative agent that supports users in their interaction with computers. Although admittedly still incomplete, the presented framework provides a simple and elegant solution to empower a computer system with a generate component that produces adequate feedback utterances at the conceptual level in human–computer interaction. The type theoretical model presented in this article seems, due to its build in property of sequentiality and its constructive nature, a valuable candidate for the type of discourse representation that enables us to mimic the process of common ground coordination. It remains for the future, however, in how far the introduced model will be sufficiently rich to take care for an adequate feedback process between humans and computer systems at the conceptual level.

## ACKNOWLEDGMENTS

We thank the editors of this special issue, Paul Piwek and Peter Kühnlein, and the anonymous reviewers for their valuable comments on earlier versions of this article.

## REFERENCES

- Agarwal, P., Huang, Y., & Dimitrova, V. (2005). Formal approach to reconciliation of individual ontologies for personalisation of geospatial semantic Web. In Rodriguez, M. A., Cruz, I. F., Egenhofer, M. J., & Levashkin, S. (Eds.) *First International Conference on GeoSpatial Semantics (GeoS 2005)* (LNCS Vol. 3,799 pp. 195–210). Heidelberg, Germany: Springer-Verlag.
- Ahn, R. M. C. (2001). *Agents, objects and events: A conversational approach to knowledge, observation and communication*. Unpublished doctoral thesis, Eindhoven University of Technology, The Netherlands.
- Ahn, R. M. C., Beun, R. J., Borghuis, T., Bunt, H. C., & van Overveld, C. W. A. M. (1995). The DenK architecture: A fundamental approach to user-interfaces. *Artificial Intelligence Review*, 8, 431–445.
- Ainsworth, W. A., & Pratt, S. R. (1992). Feedback strategies for error correction in speech recognition systems. *International Journal of Man Machine Studies*, 36, 833–842.
- Allwood, J. (1995). *An activity based approach to pragmatics* (Tech. Rep. No. 75). Gothenburg, Sweden: University of Göteborg Press.
- Allwood, J., Nivre, J., & Ahlsen, E. (1992). On the semantics and pragmatics of linguistic feedback. *Journal of Semantics*, 9, 1–26

- Baader, F., McGuinness, D. L., & Patel-Schneider, P. F. (2003). *The description logic handbook: Theory, implementation and applications*. Cambridge, England: Cambridge University Press.
- Bernsen, N. O., Dybkjær, L., & Dybkjær, H. (1996). Cooperativity in human-machine and human-human spoken dialogue. *Discourse Processes*, 21, 213-236.
- Beun, R. J., van Eijk, R. M., & Prüst, H. (2004). Ontological feedback in multiagent systems. In N.R. Jennings, C. Sierra, L. Sonenberg, & M. Tambe (Eds.), *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)* (pp. 110-117). New York: ACM Press.
- Borghuis, V. A. J. (1994). *Coming to terms with modal logic. On the interpretation of modalities in typed  $\lambda$ -Calculus*. Unpublished doctoral thesis, Eindhoven University of Technology, The Netherlands.
- Bunt, H. C., Ahn, R. M. C., Beun, R. J., Borghuis, T., & van Overveld, K. (1998). Multimodal cooperation with the DenK-system. In H. C. Bunt, R. J. Beun, & T. Borghuis (Eds.), *Multimodal human computer communication*. (Vol. 1374, pp. 39-67). Berlin: Springer-Verlag.
- Clark, H. H. (1996). *Using language*. Cambridge, England: Cambridge University Press.
- Clark, H. H., & Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, 13, 259-294.
- Gabsdil, M. (2003). Clarification in spoken dialogue systems. In *Proceedings of the 2003 AAAI Spring Symposium. Workshop on Natural Language Generation in Spoken and Written Discourse* (pp. 28-35). Stanford: Stanford University Press.
- Ginzburg, J. (2005a). Abstraction and ontology: Questions as propositional abstracts in type theory with records. *Journal of Logic and Computation*, 15, 113-130.
- Ginzburg, J. (2005b). Situation semantics: The ontological balance sheet. *Research on Language and Computation*, 3, 363-389.
- Grice, H. (1975). Logic and conversation. In P. Cole & J. Morgan (Eds.), *Speech acts. Syntax and semantics* (Vol. 11, pp. 41-58). New York: Academic.
- Hameed, A., Sleeman, D., & Preece, A. (2002). Detecting mismatches among experts' ontologies acquired through knowledge elicitation. *Knowledge-Based Systems*, 15, 265-273.
- Hirst, G., McRoy, S., Heeman, P., Edmonds, P., & Horton, D. (1994). Repairing conversational misunderstandings and non-understandings. *Speech Communication*, 15, 213-230.
- Joshi, A. K. (1983). Varieties of cooperative responses in question-answer systems. In F. Kiefer (Ed.), *Questions and answers* (pp. 229-240). Dordrecht, The Netherlands: Reidel.
- Kamp, J. A. W. (1981). A theory of truth and semantic representation. In J. Groenendijk & M. Stokhof (Eds.), *Formal methods in the study of language* (pp. 277-322). Amsterdam: Mathematisch Centrum.
- Kievit, L. (1998). *Context-driven natural language interpretation*. Unpublished doctoral thesis, Tilburg University, The Netherlands.
- Larsson, S. (2003). Generating feedback and sequencing moves in a dialogue system. In R. Freedman & C. Callaway (Eds.), *Natural language generation in spoken and written dialogue* (pp. 79-84). Menlo Park, CA: AAAI Press.
- Levinson, S. C. (1983). *Pragmatics*. Cambridge, England: Cambridge University Press.
- Luo, Z. (1994). *Computation and reasoning: A type theory for computer science* (International Series of Monographs on Computer Science). Oxford, England: Clarendon.
- McCoy, K. F. (1988). Reasoning on a highlighted user model to respond to misconceptions. *Computational Linguistics*, 14(3), 52-63.
- McCoy, K. F. (1989). Generating context-sensitive responses to object-related misconceptions. *Artificial Intelligence*, 41, 157-195.
- Nielsen, J. (1993). *Usability engineering*. San Diego, CA: Kaufmann.
- Norman, D. A. (1983). Design rules based on analyses of human error. *Communications of the ACM*, 26, 254-258.
- Piwek, P., & Kraemer, E. (2000). Presuppositions in context: Constructing bridges. In P. Bonzon, M. Cavalcanti, & R. Nossium (Eds.), *Formal aspects of context* (Vol. 20, pp. 85-106). Dordrecht, The Netherlands: Kluwer Academic.

- Purver, M. (2006). Handling clarification requests in a dialogue system. *Research on Language and Computation*, 4, 259–288.
- Ranta, A. (1994). *Type theoretical grammar*. Oxford, England: Oxford University Press.
- Rieser, V., & Moore, J. D. (2005). Implications for generating clarification requests in task-oriented dialogues. In K. Knight (Ed.) *Proceedings of the 43rd annual meeting of the ACL* (pp. 239–246). Morristown, NJ: Association for Computational Linguistics
- Rich, C., & Sidner, C. L. (1998). COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8, 315–350.
- Schlangen, D. (2004, May). *Causes and strategies for requesting clarification in dialogue*. Paper presented at the 5th SIGdial Workshop on Discourse and Dialogue, Cambridge, MA.
- Sowa, J. F. (2000). *Knowledge representation: Logical, philosophical, and computational foundations*. Pacific Grove, CA: Brooks/Cole.
- Spink, A., & Saracevic, T. (1998). Human–computer interaction in information retrieval: Nature and manifestation of feedback. *Interacting with Computers*, 10, 249–267.
- Strawson, P. (1950). On referring. *Mind*, 59, 320–344.
- Traum, D. R. (1994). *A computational theory of grounding in natural language conversation*. (Tech. Rep. No. 545). Rochester, New York: University of Rochester Press.
- van der Veer, G. C., & Puerta Melguizo, M. C. (2002). Mental models. In J. A. Jacko & A. Sears (Eds.) *The human–computer interaction handbook: Fundamentals, evolving technologies and emerging applications* (pp. 52–80). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.
- Visser, P., Jones, D., Bench-Capon, T., & Shave M. (1998). Assessing heterogeneity by classifying ontology mismatches. In N. Guarino (Eds.), *Formal ontology in information systems* (pp. 148–162). Amsterdam: IOS Press.
- Walker, M. A., Whittaker, S. J., Stent, A., Maloor, P., Moore, J., Johnston, M., et al. (2004). Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science*, 28, 811–840.

## APPENDIX

The formal rules of the type system are depicted in Figure A1.

The first rule (axiom) of the type system defines the empty context (denoted by  $\epsilon$ ) to be legal. The rules (start1) and (start2) allow us to extend a context with a statement. The rules (weakening1) and (weakening2) guarantee that context extensions are monotonic in the sense that statements remain derivable after an extension of the context. According to rules (subtype1) and (subtype2), the subsumption operator is reflexive and transitive. The rule (pred) defines the formation of predicates. The rule (apply1) states that an application of a predicate to an instance of (a subtype) of the argument type is of type “proposition.” The rule (prod) defines the formation of product types. Finally, the rule (apply2) states that an application of an instance  $f$  of a product type  $\pi x:A.P(x)$  to an instance  $a$  (of a subtype) of its argument type is an instance (i.e., proof object) of the proposition  $P(a)$ . For further information on these rules and type systems in general see, for instance, Ahn (2001) and Kievit (1998).

$$\begin{aligned}
&\varepsilon \vdash *j: \square \text{ (axiom)} \\
&\Gamma \vdash A:i \Rightarrow \Gamma, x:A \vdash x:A \text{ (start1)} \\
&\Gamma \vdash x:*s \ \& \ \Gamma \vdash y:*s \Rightarrow \Gamma, x<y:*s \vdash x<y:*s \text{ (start2)} \\
&\Gamma \vdash A:B \ \& \ \Gamma \vdash C:i \Rightarrow \Gamma, x:C \vdash A:B \text{ (weakening1)} \\
&\Gamma \vdash A:B \ \& \ \Gamma \vdash x:*s \ \& \ \Gamma \vdash y:*s \Rightarrow \Gamma, x<y:*s \vdash A:B \text{ (weakening2)} \\
&\Gamma \vdash A:*s \Rightarrow \Gamma \vdash A<A:*s \text{ (subtype1)} \\
&\Gamma \vdash A<B:*s \ \& \ \Gamma \vdash B<C:*s \Rightarrow \Gamma \vdash A<C:*s \text{ (subtype2)} \\
&\Gamma \vdash A:*s \Rightarrow \Gamma \vdash A \rightarrow *p: \square \text{ (pred)} \\
&\Gamma \vdash P:A \rightarrow *p \ \& \ \Gamma \vdash a:C \ \& \ \Gamma \vdash C<A:*s \Rightarrow \Gamma \vdash P(a):*p \text{ (apply1)} \\
&\Gamma \vdash A:*s \ \& \ \Gamma, z:A \vdash P(x):*p \Rightarrow \Gamma \vdash \pi z:A.P(z): \square \text{ (prod)} \\
&\Gamma \vdash f: z:A.P(z) \ \& \ \Gamma \vdash a:C \ \& \ \Gamma \vdash C<A:*s \Rightarrow \Gamma \vdash f(a):P(a) \text{ (apply2)}
\end{aligned}$$

where  $*j$  can stand for  $*s$  or  $*p$ ,  $i$  can stand for all types except “ $\square$ ,” and  $x$  and  $y$  are fresh.

FIGURE A1 The formal derivation rules of the type system.