# Varieties of User Misconceptions: Detection and Correction

Bonnie Lynn Webber and Eric Mays
Department of Computer & Information Science
University of Pennsylvania
Philadelphia PA 19104

## Abstract

This paper discusses some of our research into detecting and reconciling critical differences between a user's view of the world and the system's. We feel there is benefit to be gained by separating misconceptions into two main classes: misconce.pt ions about what is the case and misconceptions about what can be the case. We review some initial work in both areas and discuss our work in progress.

## 1.0 INTRODUCTION

For the past several years, we have been engaged in research aimed at extending the scope of Natural Language (NL) interaction with database systems beyond that of factual requests [Webber83], One important sub-area is that of detecting and reconciling critical differences between the user's and the system's views of the world. If not done, the result may be that the user is confused, or worse, misled by the information the system is trying to convey.

Our goal is to identify the information a system must have and use in order to detect and rectify various belief disparities, in the context of Natural Language database (db) question-answering. In this paper we review some earlier work which we now see as user misconceptions about what *is* the case in the database, as well as discussing more recent work on user misconceptions about what can be the case.

## 2.0 MISCONCEPTIONS ABOUT WHAT IS THE CASE

A user can hold many different kinds of incorrect beliefs about what is the case in the world. One type that has received initial computational attention towards its detection and correction consists of misconceptions that something exists which is describable by a particular description. If that description doesn't in fact describe anything, any question concerning additional properties true of such a thing is misguided, based as it is on a misconception. For example, consider the question

1. Which female employees work in the shoe department?

For a person to want to know the answer to this question, s/he must believe that there are some employees, that there are some female employees, and that there is a shoe department. Any one of these beliefs may be inconsistent with what is known to the system s/he is asking the question of. Recognizing and correcting such misconceptions was the aim of the CO-OP system [Kaplan79].

The problem with not doing this in responding to a user's question is the false inferences the user may otherwise draw from the answer. If the system answers "None" to the above question, the user may conclude that all female employees work in some other department, that the shoe department discriminates against female employees, etc., even though the answer may actually follow from there being no female employees or no shoe department i.e., from one of his/her "is" beliefs being wrong.

Now the type of "is"-misconception that CO-OP handles is only one of several that a user might have. For example, s/he may believe that an object has a particular attribute when it just doesn't (example 2) or that one thing depends on another when it doesn't (example 3) -

2. U: What's the maximum age for opening a Keogh account?
   S: There Is no maximum age: you can open one as long as you have income from self-employraent.

3. U: What are profit margins as a percentage of sales for each installation?
   S: Margins don't depend on sales. They are calculated as the difference between unit product cost and list price.

Such misconceptions about objects and the relationships among them is the subject of a new research effort reported on in [McCoy83].

## 3.0 MISCONCEPTIONS ABOUT WHAT CAN BE THE CASE

In addition to misconceptions about what is actually the case in the world, a user might have misconceptions about what can be the case. There are at least two types of such misconceptions. The first, given a database of entities and relations, is that some entity or subset of

entities can participate in a particular relation. As with type constraints and type violations in programming languages, this may not be the case because the entity is the wrong type. Initial work in this area is reported in [Mays 80]. The knowledge needed to recognize such type failures in users' queries consists of entity-relation information, hierarchical (subset-superset) information, as well as partition information as to what subsets of a given set are mutually exclusive. It is the last factor that is critical for distinguishing between a non-deviant request like

Which women teach courses?

and a deviant one like

Which undergraduates teach courses?

where the "teach" relation holds between "faculty" and "courses" (Figure 1). As Figure 1 shows, the entity "people" has two different partitions one between "men" and "women", the other between "faculty" and "student". Assuming a relation is always asserted at the most general point in the hierarchy, the configuration means that only faculty can be the first argument to teach, and only courses, the second. Since "faculty" and "student" have an empty intersection and "undergraduates" is a subset of "student", the implication is that "teach" cannot hold between "undergraduate" and "course". The same is not true of "women", as "women" and "faculty" can have a non-empty intersection.

Mays' system detects such misconceptions in the course of transforming a parse structure into a database query. At that point it verifies that the given arguments satisfy the constraints specified in the data model. One problem with this method is that it cannot correctly detect misconceptions in negative questions like (5).

4. Which faculty do not teach courses?
5. Which courses are not taught by faculty?

The simple check of relation/argument constraints would find both questions acceptable, and both would be translated (including negation) into database queries. Yet (5) actually reveals the user's misconception that courses can be taught by people other than faculty. (Example (4) reveals no such misconception: faculty do not have to teach courses.)

Negation has often been a source of problems for question-answering systems, but in a cleaned-up version of Mays' system, we hope to be able to deal correctly with detecting misconceptions in negative questions, as well as in positive ones.

The second type of "can be" misconception involves violating another type of constraint - constraints between events and states and their relationship over time. It is possible for a user to be mistaken about what can be true now or what could have been true (or happened) in the past, (1) because s/he is unaware of the occurrence (or non-occurrence) of some event or of its consequences or (2) because s/he believes some event has occured when it hasn't. Again, if the user's question reveals such a misconception, it should be corrected lest the user draw a false conclusion from the system's answer. The kind of behavior we are aiming for is as follows:

6. U: Is John registered for CSE220?
   S: No. He can't be registered for it because he has already advance placed it.

7. U: Is John registered for CSE220?
   S: No. He can't be registered for it because he hasn't yet taken CSE121.

The knowledge needed to recognize and square away such misconceptions consists of a knowledge of past events (or states of the dbs) — often preserved in back-up files but not accessible to the db system — and of the relationship between past events and what can be true afterwards, including possibly the present. The latter is very much like update constraints used to maintain db consistency. However, in general update constraints are not expressed in a form that admits reasoning about possible change. Something more is needed. What we have chosen to use instead is an extension of the propositional branching time temporal logic [BenAri], as documented in [Mays82,Mays831.

Our original impetus into this area was a desire to give a db system the ability to take the initiative and offer to monitor for Information of which it was currently unaware. For example,

8. U: Has John checked in yet?
   S: No - shall I let you know when he has?

9. U: Has John checked in yet?
   S: Yes - shall I let you know when the rest of the committee members do?

Work on producing monitor offers that are both competent (i.e., that correspond to a possible future state of the database) and relevant (I.e., that the user would be interested in) Is proceeding concurrently with the work reported on here. We have termed systems which can reason about possible future states of the db "dynamic database systems".

We do not have the space here to explain in detail the logical system we are using (but see [Mays83]). In brief, the system treats the past as a linear sequence of time points up to and including a reference point that, for simplicity, we can call NOW. The future is treated as a branching structure of time points that go out from (and include) the reference point. A set of complex operators Is available to quantify propositions as to the points they are asserted to hold over - e.g.,

AGq - proposition q holds at every time of every future
EXq - proposition q holds at the next point in

some future
Pq - proposition q holds at some time in the past
etc.

Two classes of axioms describe the relationship between events/states in the past, present and future. The first class contains logical axiom schemas that apply to temporal assertions in general - e.g., if prior to NOW, Pq was true (i.e., LPq), then Pq is still true NOW (i.e., LPq -> Pq). There are also "specialization" axioms relating general and more specific operators - e.g., if for all times in every future q will be true (i.e., AGq) then, more specifically, q will be true at the next time in every future (i.e., AGq -> AXq).

The second class of axioms are non-logical axioms that describe relationships that hold in the particular domain. Here we have taken a university domain of students and courses. Let the propositional letter 'a' stand for 'student advance places course' and 'r', for 'student is registered for course'. Then the following non-logical axiom states that a student who has advance placed a course (some time in the past) is not now registered for it: H[AG[Pa -> ~r]]. (Most non-logical axioms are taken to have held and to continue to hold forever. Hence the complex operator HAG around the implication.)

Our problem, given the two registration examples above (6 and 7) to distinguish whether it is accidental that John is not registered for CSE220 now (he could be, only he's not) or foreordained (some event has taken place that precludes registering or some enabling event has not yet occured) requires the system to suppress its knowledge of John's current status and consider whether it could provably believe the opposite - i.e., that John is registered now for CSE220. If it couldn't, then not only is John not registered for CSE220, the system should have identified at least one basis for why he couldn't be. By the above axiom, it is clear that it Is not accidental that John isn't registered, because r (being registered for CSE220) is inconsistent with Pa (having advance placed GSE220)
Pa -> ~r. This is the knowledge and reasoning on which we are basing the recognition of such "could be" misconceptions.

## 4.0 CONCLUSION

We have discussed recent work on detecting and correcting two main classes of user misconceptions misconceptions about what is the case and misconceptions about what could be the case, There is other research which is strongly relevant to this, in particular recent work by Mercer and Reiter on using default logic to represent the potential presuppositions of certain lexical items and syntactic constructions [Mercer82]. This is an area of great importance for interactive systems because of the critical consequences of user confusion and misunderstanding. We are continuing our efforts on it and encourage others to do so as well.

## References

[BenAri] Ben Ari, M., Manna, Z. & Pneuli, A. "The Temporal Logic of Branching Time". 8th Annual ACM Symposium on Principles of Programming Languages, Williamsburg VA, January 1981.

[kaplan79] Kaplan, S.J., "Cooperative Responses from a Portable Natural Language Data Base Query System", Ph.D. dissertation, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pa. June 1979.

[Mays80] Mays, E. "Failures in Natural Language Systems: Applications to Data Base Query Systems". Proc. 1980 National Conference on Artificial Intelligence, Stanford CA, August 1980.

[Mays82] Mays, E. "Monitors as Responses to Questions: Determining competence". Proc. 1982 National" Conference on Artificial Intelligence, Pittsburgh PA, August 1982.

[Mays83] Mays, E. "A Modal Temporal Logic for Reasoning about Change". Proc. 1983 Association for Computational Linguistics Conference, Cambridge MA, June 1983.

[McCoy83] McCoy, K. "Correcting Misconceptions: What to say when the user is mistaken". Submitted to CHI '83, Human Factors in Computing Systems, Boston MA, December 1983.

[Mercer82] Mercer, R. & Reiter, , R. "The Representation of Presuppositions Using Defaults". Proc. CSCSI-82, Saskatoon, Sask., Canada, May 1982.

[Webber83] Webber, B., Joshi, A., Mays, E. & McKeown, K. "Extended Natural Language Database Interaction". Int. J. Computers & Mathematics, Spring 1983.

Figure 1