FAILURE HANDLING IN A DIALOGUE SYSTEM

Gretchen P. Brown
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

This note focusses on the relative difficulty of handling general failure situations and failures that occur in natural language dialogue.1 The observations made here have grown out of work on a system to handle mixed-initiative task-oriented typed dialogue called Susie Software [1], which is embedded in the OWL knowledge representation system [21

Failure can be defined as a state that occurs after the violation of an expectation; a common example of a dialogue failure would be a misunderstanding. Our first observation is that dialogue failure can be just as complex as any other sort of failure. Some dialogue failures require sophisticated analytic mechanisms to isolate their causes, and in some cases even detecting a failure will require substantial effort. In addition, dialogue failure introduces some complexities of its own. For example, it is necessary to distinguish between failures occurring in dialogue and those merely reported there. On the face of it this seems an obvious enough distinction, but in analyzing dialogue protocols there is a temptation to conflate the two. Many reports of non-dialogue failures are also dialogue failures since the reports violate some of the hearer's expectations about the course that the dialogue will take. (The report may come as an interruption of, say, a question-answer sequence.) One should not conclude from this, however, that *all* reports of external failure are necessarily dialogue failures as well; to do so would be to adopt an oversimplified model that will lead to confusion.

Another special complexity is introduced by the ambiguity of language. A dialogue participant not only detects failures directly (for example, if he himself does not know the answer to a question) but also has failures reported to him by the other participant. Failures that are simple to detect when experienced directly may require more analysis when experienced secondhand because there is the additional mediation of language.

As intractable as the problem of handling dialogue failure may appear from this brief summary, our experience in work on the Susie Software system suggests that there are some areas of promise. In many cases, failure detection and recovery method selection is a straightforward process, both where failure manifestations are observed directly by an individual and where he must recognize them when reported by the other participant. Individual failures tend to have a small set of very routine signals and recovery measures. Truly novel methods of recovery from dialogue failures are rare in our experience, especially in a task-oriented environment. This is apparently due to the fact that dialogue is heavily dependent on a set of conventions understood by both parties.

Another encouraging factor is that we have found routine recovery measures for dialogue failures to be generally simple enough so that unwanted interactions with other goals are readily predictable. This does not mean that interactions are necessarily easy to avoid, just that the fact that an interaction may occur can often be detected in a straightforward manner. An example of such an interaction would be the clash between a high level goal to minimize requests to a conversational partner and the need to get more information to recover from a failure. Once a potential interaction is detected, a system can switch strategies, e.g., choose a recovery measure that requires more work but involves less interaction.

Finally, the extra mediation of language introduced in secondhand failure detection does not present as many problems as we might expect, since the failure reports that must be recognized in dialogue tend to be well marked, often with special discourse structures. Typical examples include "Do you mean ...," "I don't understand." and "Wait a minute."

The Susie Software system takes advantage of these simplifying factors in a number of different ways. First, since conventions play such an important role in failure recovery, they have been given a prominent place in the system. OWL *methods,* which are a declarative representation of procedural knowledge, are used to represent many sorts of dialogue conventions, and a special structure called a *recovery path* has been introduced to represent the more routine ways to recover from dialogue failures. Second, we distinguish different categories of failure and use several different recovery mechanisms ordered in terms of their power. This allows us to handle the simple failures simply, using more powerful mechanisms only where necessary. The implementation is also carefully designed so that failures detected firsthand and those detected through reports of the conversational partner can be modelled by similar structures.

In summary, our experience has been that, although dialogue failures can be arbitrarily complex, many exhibit properties that make them particularly amenable to handling in an appropriately designed dialogue system.

1. This work was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under Contract Number N00014-75-C-0661.

References

[1] Brown, G., A framework for processing dialogue, Technical Report 182, M.I.T. Laboratory for Computer Science, June 1977.

[2] Szolovits, P., Hawkinson, L., and Martin, W.A., An overview of OWL, a language for knowledge representation, to appear in: *Proceedings of the Workshop on Natural Language for Interaction with Data Bases,* International Institute for Applied Systems Analysis, Laxenburg, Austria.