ABSTRACT

This paper describes a system for the computer understanding of English. The system answers questions, executes commands, and accepts information in normal English dialog. It uses semantic information and context to understand discourse and to disambiguate sentences. It combines a complete syntactic analysis of each sentence with a "heuristic understander" which uses different kinds of information about a sentence, other parts of the discourse, and general information about the world in deciding what the sentence means.

It is based on the belief that a computer cannot deal reasonably with language unless it can "understand" the subject it is discussing. The program is given a detailed model of the knowledge needed by a simple robot having only a hand and an eye. We can give it instructions to manipulate toy objects, interrogate it about the scene, and give it information it will use in deduction. In addition to knowing the properties of toy objects, the program has a simple model of its own mentality. It can remember and discuss its plans and actions as well as carry them out. It enters into a dialog with a person, responding to English sentences with actions and English replies, and asking for clarification when its heuristic programs cannot understand a sentence through use of context and physical knowledge.

In the programs, syntax, semantics and inference are integrated in a "vertical" system in which each part is constantly communicating with the others. We have explored several techniques for integrating the large bodies of complex knowledge needed to understand language. We use Systemic Grammar, a type of syntactic analysis which is designed to deal with semantics. Rather than concentrating on the exact form of rules for the shapes of linguistic constituents, it is structured around choices for conveying meaning. It abstracts the relevant features of the linguistic structures which are important for interpreting their meaning.

We represent many kinds of knowledge in the form of procedures rather than tables of rules or lists of patterns. By developing special procedural languages for grammar, semantics, and deductive logic, we gain the flexibility and power of programming languages while retaining the regularity and understandability of simpler rule forms. Each piece of knowledge can be a procedure, and can call on any other piece of knowledge in the system.

Thesis Supervisor: Seymour A. Papert, Professor of Applied Mathematics