20

Issues of Representation in Conveying the Scope and Limitations of Intelligent Assistant Programs

B. G. Buchanan

Department of Computer Science
Stanford University, USA

1. INTRODUCTION

Success of a knowledge-based program depends on both competence and acceptability. It must perform well for it to be worth using, but is must be acceptable to users for it to be used. There are many dimensions to developing competent and acceptable knowledge based systems which can serve as "intelligent assistants" for problem solvers in science (see Shortliffe and Davis, 1975). One of these is the old AI problem of representation of knowledge. Since most previous work on representation has stressed its importance for problem-solving (e.g. Amarel, 1971), we will concentrate here on the importance of representation in acquiring knowledge for a reasoning program and in making the program acceptable to users.

In the past, computer assistance in science was limited to numerical calculation. Beyond numerical help, however, the computer can provide more assistance if given combinatorial and inferential abilities. That is, the problem-solving assistant should be a problem solver for the non-numeric, as well as numeric, subproblems the scientist finds tedious. For example, manipulating graph structures systematically enters into routine problems of chemistry and molecular biology: and finding the consequences of prescribing different combinations of drugs frequently enters into medical practice.

As computer programs become more complex and powerful, new problems of program design and use become apparent. Simple programs need only simple introductions. Their use is straightforward, and their output is easily interpreted. For example, a procedure that computes mean and standard deviation from a set of data requires little explanation. The most complicated step is formatting the input data. By contrast, using a complex reasoning program such as CONGEN (discussed below) requires a substantial investment of time to understand the program's scope and limitations as well as its input and output conventions. For these purposes, the choice of representation of the program's knowledge

becomes as important for communicating with users as for solving problems in the first place.

This point is illustrated in this paper by three of the DENDRAL programs, described in the next section. We briefly discuss some aspects of these programs that are relevant for the representation issues. Then in the following sections we look at two ways, in addition to problem solving, that the representation of knowledge is important for designing intelligent assistants. Communicating the scope and limits of programs is especially critical at times when a user is either augmenting the knowledge base or requesting assistance. Thus our examples focus on these two aspects. It is not necessary that the reader fully understand the details of the examples, and it is suggested that the discussion of the DENDRAL programs in the next section be skimmed on first reading.

In order to use a program intelligently, a user needs to understand the program's scope and limits. The scope, roughly, is the broad class of problems which it is designed to solve and the context in which solutions will be found. The limitations include the idiosyncrasies that must be remembered to obtain reliable solutions, but which are less fundamental to the whole procedure. For example, enumerating polymeric structures is outside the scope of CONGEN, while its working definition of aromaticity is a limitation that is more easily changed. Operationally, the scope is the broad definition of the problem which can be changed only at the cost of writing an entirely new procedure. The limitations are the explicit and implicit items in the problem definition that are added to make the problem solvable but that may be changed or removed more readily. It is not a sharp distinction; the point is that a scientist needs to understand the assistant's interpretation of the problem before the program (assistant) can be used responsibly and confidently.

2. DENDRAL PROGRAMS

The Heuristic Programming Project at Stanford University has developed a family of computer programs, collectively known as DENDRAL, which are designed to provide assistance to chemists with structure elucidation problems. In this section three of these programs are described: CONGEN, the DENDRAL Planner, and Meta-DENDRAL. The tasks that they perform are complex, even for chemists. And the depth of knowledge required for high performance forced early confrontations with problems of knowledge representation.

Many of the examples are difficult to read, and to this extent constitute negative examples to the point that input and output conventions ought to be transparent. However, we are concerned here with the more fundamental issue of conveying enough information so that the programs' reasoning framework and abilities are transparent.

2.1 CONGEN

CONGEN (see Carhart, Smith, Brown and Djerassi, 1975) is a generator of molecular structures that are consistent with constraints inferred from chemical

and spectroscopic data. (The name of the program stands for "constrained generator".) CONGEN produces a list of all plausible structural descriptions for an unknown compound with a guarantee that none has been ommitted and that there are no duplicates. The criteria of plausibility, the constraints, come from the chemist or another program called the DENDRAL Planner. CONGEN provides assistance by generating chemical graphs, displaying them, and testing them for the chemist.

CONGEN is valuable for structure elucidation problems because it provides the chemist with all and only structures that are consistent with the interpretations. The empirical formula (that is, the numbers of atoms of each type in the unknown) is a necessary piece of information; the other constraints inferred from the data are all optional. The kinds of constraints one can give to CONGEN are shown in Table 14. This list describes features of chemical molecules that can be represented as features of graphs, and thus helps delimit the scope of CONGEN.

Table 1 – Types of constraints accepted by CONGEN.

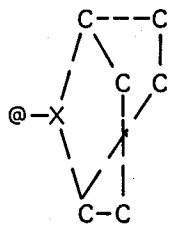
Composition:	
(a) SUPERATOMS	– Inferred polyatomic structural fragments with one or more potential bonding sites (free valences).
(b) ATOMS	– Any remaining atoms of any type, e.g., C, N, O.
Constraints:	
(c) BADLIST	– Forbidden structural features.
(d) GOODLIST	– Required structural features.
(e) BADRINGS	– Forbidden ring sizes.
(f) GOODRINGS	– Required ring sizes.
(g) PROTON	– Desired protons and their environments.
(h) ISOPRENE	– Desired isoprene units and their linkages.
(i) HRANGE	– Allowed numbers of hydrogens on specific atoms.

A convenient substructure definition language (EDITSTRUC) allows easy construction and modification of superatoms and other structural subunits the chemist wants to mention in the problem statement. An example of its use (to define a superatom named "parabridges") is shown in Fig. 1. This substructure used on BADLIST will disallow structures possessing an aromatic ring with a bridge of fewer than eight atoms between atoms that are opposite ("para" to) one another. Although this language is not self-documenting and requires brief instruction, the commands are easily understood and remembered by

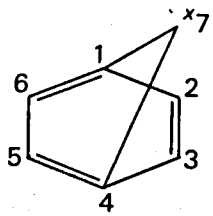
chemists. They are designed to be terse enough to be fast; for example, repeated calls to the same command can be avoided by typing additional sets of arguments into the first call.

```
   [editstruc]
NAME:[parabridges]
(NEW STRUCTURE)
>[ring 6]
      ; Start with a six membered ring.
>[bord 1 2 any 2 3 any 3 4 any 4 5 any 5 6 any 6 1 any]
      ; Allow any bond order between atoms 1&2, 2&3, 3&4, etc.
>[artype 1 a 2 a 3 a 4 a 5 a 6 a]
      ; Require all atoms to be aromatic.
>[link 1 4 1]
      ; Link atoms 1 & 4 with a new atom.
>[lnode 7 1 7]
      ; Allow the new linking atom (7) to be a chain of one to
        seven atoms.
>[atname 7 x]
      ; Let atom 7 be any atom type (x).
>[adraw]

PARABRIDGES: (AROMATIC ATOMS AND "ANY" BONDS NOT
INDICATED)
LNODES ARE INDICATED BY AN ATTACHED @
```



[A chemist would reconstruct this as:



where node 7 (*X*) stands for a bridge containing 1-7 atoms.]

Fig. 1 – Definition of a superatom for CONGEN with the structure editing language EDITSTRUC. [Bracketed characters typed by use. Annotations follow a semi-colon.]

The more constraints there are, naturally, the smaller the list of structures will be. For example, the program will produce 284 structural isomers with the composition C6H13N, but will generate only 4 if the program is told there is one six-membered ring and one methyl group. More complex examples of CONGEN problems are given by Carhart *et al.* The size and complexity of problems that CONGEN can handle are limited more by the number of structural units (super-atoms plus remaining atoms) than by the total number of atoms in the empirical formula. Thus CONGEN can help with structural problems of real interest when the chemist is able to infer some structural features from available data.

2.2 DENDRAL Planner

The DENDRAL Planner (see Smith *et al.*, 1972) is a program designed to aid in the interpretation of mass spectra. It is not "automatic" in the sense of producing structural interpretations of an unknown mass spectrum with no intervention from the chemist. Instead, it uses the chemist's relevant knowledge of mass spectrometry and applies it systematically to the spectrum of an unknown. That is, using the chemist's definitions of the structural skeleton of the molecule and the relevant fragmentation rules, the program does the bookkeeping of associating peaks with fragments and the combinatorics of finding consistent ways of placing substituents around the skeleton.

Fig. 2 shows the structure of the common skeleton (defined with EDITSTRUC) for a class of compounds called capnellanes. For purposes of illustration, three major fragmentations were defined, which are shown schematically in Fig. 2, overdrawn on the computer's drawing of the skeleton. It is possible to specify other definitions of the context in which the chemist wants the interpretation to be made, such as the intensity threshold for noise peaks.

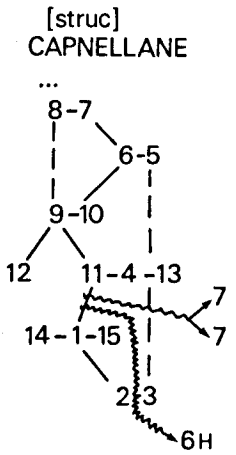


Fig. 2 – A structural skeleton defined for the DENDRAL Planner. Fragmentations were manually drawn on the computer's output. [Bracketed characters typed by user.]

From the information about the skeleton and fragmentations for the general class, the program begins interpreting the mass spectrum as shown in Figs. 3 and 4. First the empirical formula (or "molecular ion") is determined (see Dromey, Buchanan, Lederberg, and Djerassi, 1975). Then it finds data points (peaks) in the spectrum that could plausibly be associated with the defined fragmentations, with and without combinations of substituents. (The range of possible substituents is determined by subtracting the composition of the skeleton from the empirical formula for the determined molecular ion.)

```
[plan]
(COMPUTING MOLECULAR ION(S))

MOLECULAR IONS
(234. 1609 100 (C . 15) (H . 22) (0 . 2))

    ;Only one plausible molecular ion peak is found for this
    problem, at mass 234.1609 and intensity 100.
    ;From the exact mass 234. 1609 only one empircal formula is
    plausible, viz, C15 H22 O2.
```

Fig. 3 – Start of planning: molecular ion determination. Annotations follow semicolons.

```
(STARTING ANALYSIS PART)
BREAK : SUBSTITUENTS ON CHARGED FRAGMENT : EVIDENCE (M/E)
6H    ((DOT . 4) (C . 0) (O . 1))     161.0975
      ((DOT . 2) (C . 0) (O . 1))     163.1136
      ((DOT . 4) (C . 0) (O . 2))     178.0993

;The format is not clear as it should be. There are three sets of alternative
substituents on the fragment resulting from Break 6H (see Fig. 2).
These are (a) 2 double bonds or rings (4 "dots") and one oxygen,
(b) 1 double bond or ring and one oxygen, or (c) 2 double bonds or rings
and two oxygens.
Each has supporting evidence in the data at the masses shown,
-----------------------------
7H    ((DOT . 4) (C . 0))             133.1009
      ((DOT . 2) (C . 0))             133.1009
      ((DOT . 4) (C . 0) (O. 1))      150.1038  149.0951  148.088
                                      147.0811
      ((DOT . 2) (C . 0) (O . 1))     150.1038  149.0951
      ((DOT . 4) (C . 0) (O . 2))     163.0758
-----------------------------
7L    ((DOT . 4) (C . 0))             65.03971
      ((DOT . 2) (C . 0))             67.05513
      ((DOT . 0) (C . 0))             69.07053
      ((DOT . 0) (C . 0) (O . 1))     85.06461
-----------------------------
```

Fig. 4 – Intermediate planning results: likely assignments of special peaks to fragmentations. Fragmentations 6H, 7H, and 7L are shown in Fig. 2. Annotations follow semicolons.

The output from the Planner is a list of structure descriptions with as much detail filled in as the data and defined fragmentations will allow. Because there are limits to the degree of refinement allowed by mass spectrometry alone, sets of atoms are assigned to sets of skeletal nodes. Thus the task of fleshing out the plan – specifying possible structures assigned to specific skeletal nodes – is left to CONGEN. Figure 5 shows the program's output for the capnellane example: there is only one structure, and it has one oxygen and two double bonds (or extra rings) within nodes C4-C13, with one oxygen on node C3. (Most probably, these are a keto [C=O] group and carbon-carbon double bond somewhere within C4-C13 and hydroxyl [OH] on C3.) This partial description of the unknown can be used by CONGEN, together with other constraints, to produce a list of complete structures.

```
        BEGIN SYNTHESIS OF MOLECULAR ION = 234.1609
STRUCTURE 1
(((DOT . 4) (O . 1)) C4 C5 C6 C7 C8 C9 C10 C11 C12 C13)
(((O . 1)) C3)

     ;The evidence shown in Figures 4 can be consistently
     combined in only one way.

     ;i.e., two double bonds or rings (four "dots")
     and one oxygen atom can be placed within nodes 4-13 of
     the skeleton, and one oxygen is on node 3.

EVIDENCE USED TO BUILD STRUCTURE:
(6H (DOT . 4) (O . 2))
(7H (DOT . 4) (O . 1))
(7L (O . 1))

     ;The parens and dotted pairs are not helpful,
      but the information helps the chemist relate
      final conclusions to the intermediate reasoning.

DONE
```

Fig. 5 – Results of DENDRAL Planner: description of structure(s) consistent with the data. Annotations follow semicolons.

2.3 Meta-DENDRAL

The Meta-DENDRAL program (Buchanan *et al.*, 1976) is designed to provide assistance to mass spectroscopists who are formulating new fragmentation rules to explain the behaviour of a new set of compounds. It begins with a collection of known structure-spectrum pairs. From these data, together with the chemist's defined criteria of plausibility, the program finds plausible fragmentations and rearrangements that account for many of the most significant peaks in the spectra. For example, rules 6H, 7H, and 7L used in the example above (Figs. 2 to 5) can result from the program's examination of a training set of capnellane structures and their known mass spectra.

The mass spectrometry rules are written in terms of a topological description of a piece of a molecule (a subgraph) and a corresponding fragmentation or rearrangement process. For example, rule M-6 in Fig. 6 says that in the presence of a keto group we would see evidence for fragmentation of the bonds opposite and adjacent to that group. The subgraphs are described in "ball and stick" terms, with no stereochemistry. The corresponding processes are defined in terms of bond cleavage(s), net transfer(s) of hydrogen, or neutral species such as water, and charge placement. Because the program writes new rules in terms of an existing vocabulary (and does not invent new terms) it is extending an existing theory but not developing a new one.

Two rules produced by the program are shown in Fig. 6 along with a summary of the evidential support for them in the training set. These are taken from a results table of a paper (see Buchanan *et al.*, 1976) describing the program's help in formulating mass spectrometry rules for three classes of compounds not previously codified in this way (the mono-, di-, and triketoandrostanes).

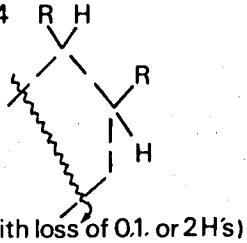
Name	Subgraph	Positive Evidence Any	Positive Evidence Unique	Negative Evidence	Average Intensity Percent
. . .					
M-4	 (with loss of 0,1, or 2 H's)	21	0	1	3.71
M-6	 (with loss of one H)	8	4	0	3.29

Fig. 6 – Two general fragmentation rules found by Meta-DENDRAL. (The positive evidence is the number of data points correctly predicted by the rule: it is unique if no other rules also predict the same data point.
Negative evidence is the number of peaks predicted by the rule but not found in the data.

The rules are formed in three distinct stages: (1) data interpretation and summary, (2) rule generation, and (3) rule modification. Each depends on the chemist's definitions of the context in which rules will be formed. For example, the context includes whether the program should consider cleavage of aromatic ring bonds and multiple bonds, the numbers of hydrogens to consider in rearrangements, the types and numbers of neutral fragments (such as water) to be lost, and the complexity of the processes that can count as explanations of peaks. A sample of the interaction specifying the context is shown in Fig. 7.

```
specify fragmentation process constraints? Y/N. : [Y]

prompt for all constraint values? Y/N. : [Y]
forbid cleavage of more than one bond
     to the same atom? Y/N. : [Y]
forbid cleavage of aromatic ring bonds? Y/N. : [N]
allow default definition of aromatic rings? Y/N. : [Y]
forbid cleavage of double and triple bonds? Y/N. : [Y]
minimum number of carbons in a fragment? : [2]
allowed hydrogen transfers? : [2 1 0 −1 −2]
maximum number of bonds allowed to cleave
     in a single step process? : [2]
maximum number of steps in a fragmentation process? : [2]
maximum number of bonds allowed to cleave
     in a multiple-step process? : [2]
maximum number of rings allowed to fragment
     in a multiple-step process? : [1]
allowed neutral transfers (other than H)?
     : [CO −1]
any other constraints? Y/N. : [N]
```

Fig. 7 – A sample of the context definition for Meta-DENDRAL. [Bracketed characters were typed by user.]

The rule formulation procedure is not modelled after human procedures, but is perhaps more systematic and thorough than the creative methods of scientists. At bottom, it is a systematic exploration of the space of more and more specific rules that can explain the interpreted data. This is followed by a final "fine tuning" of the rules to make them more general, when possible, or more specific (if this helps reduce the negative evidence), or to merge similar rules into a common form (see Buchanan *et al.*, 1976, for details).

3. REPRESENTATION PROBLEMS IN KNOWLEDGE ACQUISITION AND VERIFICATION

Accumulating knowledge of a domain in a form that a program can interpret and use is essential for the program to assist with problems of reasoning. This is one of the central themes of all the work at the Stanford Heuristic Programming Project. We have experimented with various ways of adding scientific and

medical knowledge to programs. The three main methods that we have used we call handcrafting, interactive dialog, and automatic rule formation. The work has been performed by several persons at Stanford in the context of work on DENDRAL and other programs.

In each case we are concerned with giving the program the same kind of knowledge that an expert in the domain uses for problem solving. We do not pretend to be able to make the problem-solving assistant as sophisticated as an expert in all respects. But we want to transfer enough knowledge into the program to make it a useful assistant.

3.1 Functional components

In order to build a high performance reasoning program four logically separable systems must interact. These are

(1) an expert,
(2) a transfer agent,
(3) a reasoning program,
(4) a verifier.

In most instances, AI programs are built by single individuals who act as expert, transfer agent and verifier, very often in domains that require no more formal source of knowledge than common sense. Many interesting specialized systems have recently been developed which rely on much more than common sense. Some programs are constructed by persons who are first and foremost domain experts (for example, Colby in psychiatry, Berliner in chess); others are constructed by programmers who become experts in a specialized area of science (for example, Reddy in speech understanding). However, even in cases where an individual fills the role of two or more of the four systems, it is instructive to separate the activities of the component parts.

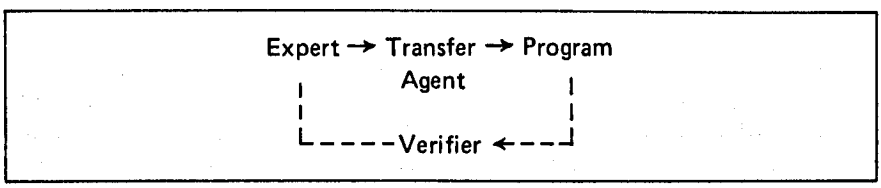


Fig. 8 – Functional components for building an expert system.

3.1.1 *Expert*

For most system building activities we think of the expert as a human being. In this decade, a person is far more effective in this role than a program would be, because of the breadth and depth of knowledge required for teaching complex tasks. (See Waterman, 1970, for comparisons of both a human and a program as expert.) Representation problems become critical in knowledge

acquisition and in verification, since the expert is expected to match what the program can and cannot do.

Although we often think of the domain expert as the ultimate source of expertise for AI programs, we can also consider the origin of that individual's expertise as a source of knowledge. In science there is one ultimate source – observations, or empirical data. Experts also learn from textbooks and papers written by others who have codified and explained empirical observations.

3.1.2 *Transfer Agent*

There is no reason to expect the expert to know the details of the programming language and the implementation of the reasoning program. Perhaps it is more accurate to say that the expert's own codification of knowledge is not executable as a program without some translation. The expert sometimes serves as translator, but usually this task is given to someone closer to the program itself.

Transfer can also be accomplished by a program. See Davis, 1976, for a fuller discussion of interactive knowledge transfer issues and methods for aiding the transfer. Meta-DENDRAL avoids the link with the expert by formulating rules directly from the original data and transferring them to the performance program.

3.1.3 *Program*

There are many examples of expert systems which illustrate the kind of performance we expect from a program built under the supervision of an expert. DENDRAL and Meta-DENDRAL are two of these.

For the cost of eliciting knowledge from experts, we expect the resulting program to be superior to one developed without the aid of an expert. Primarily this means that we expect superior performance. In addition, however, we want the program to be easier to understand, easier to justify, and easier to modify.

3.1.4 *Verifier*

In addition to the performance program, we, as system builders, need mechanisms for determining the program's level of expertise. Programmers are used for correcting syntactic errors and verifying that the program meets performance requirements on test cases. If we were translating algorithms into programs, almost all of the verification could be done this way. However, in a knowledge-based AI program there may be errors in the knowledge base that are impossible for anyone but an expert to detect and correct. For this reason the feedback is shown to the expert in Fig. 8, not to the transfer agent. For this reason also, the output of the reasoning program and its intermediate conclusions must be easily understood by the expert.

3.2 Representation issues

The choice of representation for the expert's knowledge must be a compromise between what is natural and easy for the expert and what is natural and easy

for the programmer (transfer agent). The choice of concepts and relations must certainly come from the expert, but the means of encoding them be partly chosen by the programmer. For example, the DENDRAL representation of chemical molecules as two-dimensional connected graphs was chosen by Professor J. Lederberg as an appropriate representation for mass spectrometry problems. The actual implementation of graph theoretic concepts and operations in LISP lists, and operations on lists, was based more on programming convenience than on chemical considerations.

However, there is no reason to believe that the expert's first conceptualization is complete or correct. The expert will begin with classical textbook statements that are reasonably certain and well accepted. But in domains like mass spectrometry and medicine there is still a gap between what the textbooks say and what is needed for expert performance. The differences can show up in many ways. For example, the textbooks may lack statements of relations that are less than certain but are used in practice; they may fail to make distinctions that are useful in practice, and they may state facts at a theoretical level not used in practice. In addition, as the field grows, the textbooks lose their claim to completeness.

Similarly, there is no reason to believe that the programmer's first implementation is correct and appropriate. There is tremendous potential for misinterpreteing what the expert says when the programmer has little understanding of the domain. A more serious problem is that the internal representation chosen by the programmer, on the basis of initial conversations with the expert, will be inappropriate for ideas to be incorporated later.

Overcoming this communication barrier is a major hurdle in constructing a knowledge-based system. Both the expert and the programmer are simultaneously developing representations of the domain that they believe are appropriate for the task and for the program that performs that task. Thus it is not surprising that new applications programs often need to be completely rewritten after their initial implementations.

Figs. 1 and 2 show parts of the knowledge acquisition processes for DENDRAL programs. These have been designed with an experienced user in mind – one who has read the documentation, but not necessarily one who understands the details of the program. The knowledge built up in these ways can be saved and used in other problems. The TEIRESIAS system (Davis, 1976) emphasizes interaction with naive users much more than DENDRAL's knowledge acquisition procedures. The main point is still the same: limits on what the program can represent are limits on what knowledge it can acquire.

The programs' conceptualization of the domain of chemistry was moulded by chemists. It does not include everything chemists know about molecular structures or mass spectrometry, but it meshes well with a subset of their knowledge. The most significant difficulties come from mismatches between how chemists think about part of a problem and how the program represents that part. For example, CONGEN treats double bonds as rings of size two,

for reasons of efficiency and completeness, yet chemists often make a clear distinction between rings and double bonds, for chemical reasons. Thus CONGEN has been modified in many places to recognize the distinction even though it is not necessary for problem solving. Without mapping the program steps into the chemist's conceptual framework, there would be great difficulty in verifying and augmenting the knowledge base.

4. REPRESENTATION ISSUES IN USE AND ACCEPTANCE

For a high-performance computer program to capture the sustained, widespread attention of working scientists, it must contain a large number of features that make it easy and pleasant to use. These features are commonly termed "human engineering aspects" of a program. In very rare instances, a program will be so useful that it will be widely adopted even without proper attention to human engineering. But the general principle seems to be that programs that are only understandable to programmers are used only by programmers, if at all.

Just as the expert creating the program must feel that the representation of knowledge is appropriate for the task, the user of the system must also feel comfortable with it. This is an indispensable element of acceptability: problem solutions must be presented to the user in a familiar vocabulary. A common error we make is to present solutions in the vocabulary of the programmer, using the concepts of the program instead of those of the domain practitioners and experts. (Even though the output shown in Fig. 5, for example, could be improved with respect to notation and readability, the concepts are the right ones to convey.) Occasionally one might find cases where the expert's conceptualization is not fully shared by the users, in which cases some further translation is also required for the users.

Because an AI program can be a powerful tool it also has the potential of powerful misuse. Paraphrasing a message by J. Weizenbaum (1976) (to teachers of computer science), the designers of intelligent programs have a responsibility to teach users the limitations of their tools as well as their power. A most important set of limitations stems from the choice of representation of knowledge about the problem domain. Whatever representation is chosen, it will have its own set of limitations and peculiarities that can mislead and confuse the users of the program. For example, users who expect DENDRAL to give them information about bond lengths and angles in its drawings of molecular structures are going to misinterpret those drawings.

4.1 Documentation and assistance

A designer usually leaves to the user the task of understanding a program's scope and limitations by telling him that he must read and understand the program documentation before beginning. But documentation usually does not discuss appropriate uses of a program or the program's limitations. (Unfortunately, the more complex a program is and the more a user needs full documentation of its scope and limits, the less willing programmers are to create the document).

While it is tempting to think of intelligent use of a program in terms of understanding its documentation, the program itself can aid considerably. One of the exciting possibilities of the future is creating intelligent assistants that carry an awareness of their own problem-solving abilities and can explain them to users.

Documentation is the most obvious way of conveying the scope and limitations of a program. We are beginning to work with some of the easier steps of learning about a program and to shift some of the burden from documentation to interactive aids built into the program. For example, the programs themselves can be asked what to do next, what the available commands are, what a prompt means, what the effect of a parameter is. Some of these features are illustrated in the context of the DENDRAL Planner. Fig. 9 shows the program's response to a request for a summary of available commands. Fig. 10 shows the program's descriptions of two critical parameters.

```
[menu]
*** MENU OF AVAILABLE COMMANDS ***
COMMAND                 WHEN-APPLICABLE     PURPOSE
?                       ANYTIME             TO GET MORE INFORMATION
CNTRL-E                 ANYTIME             TO CANCEL COMMAND
GET                     AFTER >             TO GET PARAMETERS FROM A
                                            CLASS FILE
<PARAMETERNAME>         AFTER >             TO DEFINE NEW VALUE FOR
                                            PARAMETER
<PARAMETER>?            AFTER >             TO SEE THE VALUE OF A PARAMETER
READ                    AFTER >             TO READ A MASS SPECTRUM FROM A
                                            FILE
PLAN                    AFTER >             TO RUN THE PLANNER PROGRAM
SAVE                    AFTER >             TO SAVE THE DEFINED PARAMETERS
DESCRIBE <COMMAND>      AFTER >             TO SEE A DESCRIPTION OF THE
                                            COMMAND
LIST                    AFTER >             TO SEE IMPORTANT PARAMETER
                                            NAMES
DESCRIBE <PARAM>        AFTER >             TO DESCRIBE THE NAMED
                                            PARAMETER
DONE                    AFTER >             TO EXIT FROM THE PROGRAM
```

Fig. 9 – The DENDRAL Planner's description of commands. [Bracketed characters were typed by user.]

Through documentation or otherwise, it is necessary to convey to the chemist what the program knows about the current problem. This is harder as more knowledge is put into the program. We have tried to keep the judgmental knowledge used by DENDRAL programs separate from the coded procedures that depend on that knowledge. In this way we can, in principle, explain what each program knows. In CONGEN, for example, we give the chemist control over (almost) all the specifications of the structure elucidation problem. The

constraints are defined and modified by the person with the immediate problem, as illustrated in Fig. 1. The whole set of constraints can be examined on-line in the course of problem solving, as shown in Fig. 11.

```
[list]
   *** ESSENTIAL PARAMETERS ***
>>> STRUC
"STRUC (PARAMETER) IS THE STRUCTURE OF THE BASIC SKELETON OF
THIS CLASS. IT MAY BE EDITED BY USING THE FOLLOWING COMMAND:
   EDIT STRUC
THIS COMMAND INVOKES THE STRUCTURE EDITING PROGRAMS USED
IN CONGEN AND OTHER DENDRAL PROGRAMS."

>>> BREAKS
"BREAKS (PARAMETER) IS THE SET OF FRAGMENTATION RULES FOR
THE SKELETON; EACH BREAK DESCRIBES THE BONDS BROKEN AND
THE HYDROGENS TRANSFERRED"
```

Fig. 10 – Example of program's description of its parameters. [Bracketed characters were typed by user.] Programs with the same names as the parameters are available to aid in defining new values.

```
[contraints?]
-------
BADLIST CONSTRAINTS
   NAME
PARABRIDGES
-------
GOODLIST CONSTRAINTS
   NAME  MIN MAX
CH-3       1   1
-------
```

Fig. 11 – CONGEN's response to request for description of current constraints. Parabridges is shown in Fig. 1; other superatoms would be defined similarly.

There remains much subtle knowledge of chemistry embedded in the programs that we are not able to display, except through hard-copy documentation. For example, Meta-DENDRAL uses "built in" criteria to decide when an emerging rule is good enough or when one rule is better than another. And CONGEN cannot explain to the chemist how subproblems are ranked to determine their order of solution, or how problem size is estimated. Perhaps the closest we have come to this is in the DENDRAL Planner, where the user can see and change the complex conditions under which the program assigns evidence to fragmentations as shown in Fig. 12.

```
[controlrules]
USE ONLY THE STRONGEST EVIDENCE FOR SOME BREAKS (Y/N)?[y]
APPLICABLE BREAKS: [6h]
PASSES: [1]
USE EVIDENCE THRESHOLD FOR SOME BREAKS (Y/N)?[y]
APPLICABLE BREAKS: [7h 7l]
APPLY ON PASSES: [1]

PERCENT OF MAXIMUM INTENSITY (DEFAULT = 33): [?]

WHAT THRESHOLD DO YOU WANT AS A CUTOFF – GIVE A NUMBER THAT
WILL BE USED AS PERCENT OF MAXIMUM INTENSITY TO THROW AWAY
(RELATIVELY) SMALL INTENSITY PIECES OF EVIDENCE.
PERCENT: [33]

; This requires the chemist to understand that different
  criteria can be applied to the evidence in successive attempts
  ("passes") to find structures consistent with the evidence.
  Only the strongest evidence for Break 6H, and only evidence
  above 33 percent of the maximum intensity for 7H and 7L, will be
  used on the first pass, in this example.
```

Fig. 12 – Example of user control over the DENDRAL Planner's procedure that selects evidence for fragmentations (see Fig. 2 for drawings of 6H, 7H and 7L). [Bracketed characters were typed by user.]

4.2 Laboratory notebook

With a simple program, there is little point in asking for a detailed record of progress: between problem specification and solution there is little to note. As soon as a program is expected to behave as a problem-solving assistant on complex subproblems, however, progress notes printed by the program take on the importance of a laboratory notebook.

There are three different kinds of notes we expect the DENDRAL programs to provide:

(a) a record of initial conditions, intermediate conclusions, and final results;
(b) a complete record of the interaction between chemist and program (including false starts and typing mistakes);
(c) a trace of the program's reasoning steps.

Each of these is important for a different reason. The final results, of course, are the *sine qua non* of the assistant's work. The record of initial conditions and major intermediate conclusions gives the chemist at a glance the context in which the problem was solved and the major steps in its solution. This serves as a useful reminder of scope and limits; in addition, disagreement on initial conditions or intermediate conclusions would be sufficient reason to

request the assistant to start over. Meta-DENDRAL, for instance, immediately precedes its stored and printed results with a summary of the context specified by the chemist. Because they are together there is less chance that the results will be interpreted without proper regard for the context.

The record of the chemist's interaction with the program is a detailed account of what the investigator requests of the assistant. Failure to find a solution to a problem can often be attributed to ill-specified requests, so it is helpful to review the complete record of specifications made by the investigator. The requests for help and the program's response illustrated in the previous section are important entries in the experimental record.

Finally, the trace of the assistant's reasoning steps is helpful whenever the investigator wants to keep track of the inferential steps of an assistant. In any case it is often useful to have a record in order to justify moving from one point to the next. For example, before the DENDRAL Planner prints the results shown in Fig. 5 above, it prints the plausible molecular ions it inferred from the data (Fig. 3) and the data it associates with each of the separate fragmentations (Fig. 4).

In all cases, the entries in the laboratory notebook must be easily interpreted by the investigator for them to be useful. We have much to learn in this respect. The examples above are still cryptic and hard for outsiders to understand. But within each entry there is much information of value to a chemist about the program's problem solving procedure.

4.3 Multiple representations

One thing that we can do to help the chemist discover the scope and limits of the program's problem-solving abilities is to present information to the chemist in more than one way. Our representation of chemical structures is very limited. We do not know how to convey its limitations adequately, although we do present structures in several ways in order to give the chemist as good a feeling for the limits of the representation as we can. The documentation is explicit in these matters: the programs themselves are usually not as explicit. In CONGEN we present structures as pictures of graphs, as connection matrices, as descriptions of graphs with node properties listed. From the collection of these representations we hope that a chemist will be able to see accurately what the program assumes about molecular structures, even if he avoids reading the documentation. Fig. 1 above showed the definition of a superatom and the program's rough drawing of it. Another description of the superatom (for experienced users) is shown in Fig. 13.

The scope of Meta-DENDRAL is conveyed in similar ways. For example, the program works with mass spectrometry fragmentaion rules with possible hydrogen migration and loss of neutral species. By describing those processes in different ways, we expect the chemist to see, for example, that the program knows only about net losses.

```
>[show]
NAME=PARABRIDGES
ATOM  TYPE  ARTYPE   NEIGHBORS  LNODE
  1     C     AROM     7  6  2
  2     C     AROM     1  3
  3     C     AROM     2  4
  4     C     AROM     7  3  5
  5     C     AROM     4  6
  6     C     AROM     5  1
  7     X     NON-AR   1  4      1-7
BONDS 6-1, 5-6, 4-5, 3-4, 2-3 AND 1-2 ARE OF TYPE "ANY"
>[done]
```

Fig. 13 – Another description of the superatom in Fig. 1.

4.4 Description of context

The context in which problem solving proceeds is essential information for interpreting the solutions. The more an assistant can make explicit the assumptions and initial conditions of a problem, the easier it is for an investigator to understand the answers. This has always been true, but the emergence of computer programs as assistants brings the problem clearly into focus.

In a program the assumptions are often completely hidden. As computer programs become able to convey many of their own assumptions and limitations, however, the users will be able to delegate significant parts of their problems with confidence they will be solved as the user wants them solved.

The only step we have made along these lines with DENDRAL programs is to keep a good laboratory notebook, as described above. One of the items we try to make explicit at the time problem solutions are printed is the set of assumptions under which the program arrived at those solutions. Much more remains to be done.

5. CONCLUSION

We do not want to inflate our expectations of future programs to the point that we are bound to be disappointed. On the other hand we cannot remain content with computer programs in which the whole burden of intelligent use is placed on the user. Computer programs will have to contain much more knowledge of the domain and of their own procedures in order to make significant differences in the practice of science.

It is essential to find representations of the knowledge contained in programs that are appropriate for the persons using them and for the experts who help build them. Clever representations for expert problem solving are only half the story of creating intelligent assistants that aid working scientists. Equally important are the representations commonly used in human practice that allow users to determine the scope and limitations of what their assistants can do.

Acknowledgements

These ideas were developed in collaboration with numerous members of the Heuristic Programming Project at Stanford University. I am particularly grateful to Drs. Dennis Smith, Randy Davis, Robert Engelmore, and Tom Mitchell for their help. This work was supported, in part, by the National Institutes of Health (Grant RR-00612) and the Advanced Research Projects Agency (ARPA Order DAHC 15-73-C-0435).

REFERENCES

Amarel, S. (1971). Representation and modelling in problems of program formation, *Machine Intelligence 6*, pp. 165-190 (eds. Meltzer, B. and Michie, D.). Edinburgh: Edinburgh University Press.

Buchanan, B. G., Smith, D. H., White, W. C., Gritter, R., Feigenbaum, E. A., Lederberg, J. and Djerassi, C. (1976). Applications of artificial intelligence for chemical inference XXII. Automatic rule formation in mass spectrometry by means of the Meta-DENDRAL program. *Journal of the American Chemical Society*, 98, 6168-6178.

Carhart, R. E., Smith, D. H. Brown, H. and Djerassi, C. (1975). Applications of artificial intelligence for chemical inference XVII. An approach to computer-assisted elucidation of molecular structure. *Journal of the American Chemical Society*, 97, 5755-5762.

Davis, R. (1976). Applications of Meta-Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases, Ph.D. thesis in Computer Science, Stanford University.

Dromey, R. G., Buchanan, B. G., Lederberg, J. and Djerassi, C. (1975). Applications of artificial intelligence for chemical inference XIV. A general method for predicting molecular ions mass spectra. *Journal of Organic Chemistry*, 40, 770-774.

Shortliffe, E. H., Davis, R. (1975). Some considerations for the implementation of knowledge-based expert systems, *SIGART Newsletter*, 55, 9-12.

Smith, D. H., Buchanan, B. G., Engelmore, R. S., Duffield, A. M., Yoe, A., Feigenbaum, E. A. Lederberg, J. and Djerassi, C., (1972). Applications of artificial intelligence for chemical inference VIII. An approach to the computer interpretation of the high resolution mass spectra of complex molecules. Structure elucidation of estrogenic steroids, *Journal of the American Chemical Society*, 94, 5962-5971.

Waterman, D. A. (1970). Generalization techniques for automating the learning of heuristics, *Artificial Intelligence*, 1, 121-70.

Weizenbaum, J. (1976). *Computer Power and Human Reason*. San Francisco: Freeman.