Computer-Aided Reasoned Discourse, or, How to Argue with a Computer

Paul Smolensky
Barbara Fox
Roger King
Clayton Lewis

CU-CS-358-87

Computer-Aided Reasoned Discourse,
or,
How to Argue with a Computer

Paul Smolensky, Barbara Fox,
Roger King, and Clayton Lewis

CU-CS-358-87 February, 1987

Departments of Computer Science and Linguistics, and
Institute of Cognitive Science
University of Colorado
Boulder, CO 80309–0430

Abstract

Reasoning is a demanding intellectual task that is greatly facilitated by appropriate notational systems. We propose to develop a new notational system for supporting *reasoned discourse:* the construction, communication, and assessment of reasoned arguments about non-formal domains. The notation we propose is an active system: a computer environment called EUCLID. In EUCLID, arguments are displayed in tabular and graphical forms; a library of argument types provides users with tools for constructing arguments, and EUCLID's knowledge of argument structure offers users tools for argument comprehension and assessment. EUCLID rests on a language called ARL for formally representing argument structure; argument content is expressed in natural language. The development of EUCLID includes research on:

- the structure of actual examples of reasoned discourse;
- the representation of argument structure using a formal language;
- the management and modeling of large data structures representing arguments; and
- the nature and effectiveness of human reasoning when the new notational system is used.

To appear in: R. Guindon (Ed.),
Cognitive Science and its Implications for Human–Computer Interaction,
Lawrence Erlbaum.

Spoken language, writing, and mathematical notation and proof are symbolic systems that have profoundly affected human reasoning capacity. Modern computers are powerful, active symbolic systems with the potential, we believe, to provide significant further advances in human reasoning ability.

In this report, we describe a tool we are developing for helping people create and assess reasoned arguments and communicate these arguments to others. The tool provides reasoners with a *language*, ARL, for expressing their arguments in a clear, precise, and relatively standardized fashion. The medium in which this language is realized is a computer environment we call EUCLID.

In EUCLID, the computer plays a role analogous to acoustic or print media in verbal or written argumentation: it provides a medium—an extremely powerful one—for supporting logical discourse among human users. We are *not* proposing to use computer reasoning to replace human reasoning. Our goal is to give users the expressive and analytic power necessary to elevate the effectiveness of their own reasoned argumentation.

In this document we offer an argument for the potential value of the EUCLID project. The top-level structure of our argument is summarized in Figure 1, which also serves as a table of contents. We begin in section 1 by elaborating the goal for the EUCLID project, and describing the capabilities ARL and EUCLID are ultimately intended to offer. Next, in section 2, we argue that a system with the capabilities we strive for can significantly enhance the effectiveness of human reasoning. This includes an analysis of the means by which notational systems amplify cognitive abilities. Then, in section 3, we discuss other research that we can build upon; this also allows us to clarify our objectives by relating them to those of other projects. We focus on this issue because we have found when discussing this research with colleagues that our goals are frequently misunderstood.

In the major portion of the report, section 4, we describe the techniques that we are employing in developing EUCLID. The proposed research can be usefully divided into four parts. The first part, considered in section 4.1, is a task analysis in which argumentation is analyzed to elucidate those aspects of argumentation for which people need most support. The second part, described in section 4.2, is the development of ARL, a language which has sufficient expressive power to represent realistically sophisticated arguments, and which explicitly offers the support that the task analysis shows people need. The third part, considered in section 4.3, is the development of EUCLID, a computer environment that provides the necessary support for effectively using ARL. The final part, considered in section 4.4, is the evaluation of the effectiveness of the evolving EUCLID system.

These four research components draw primarily upon techniques from the research areas of the three principal investigators. The task analysis is an application of discourse analysis to the special discourse type we call *reasoned discourse*. ARL is an application of artificial intelligence knowledge representation techniques to knowledge of informal logic. EUCLID can be viewed as an application of semantic database modeling to a special database with the semantics of argumentation. Evaluation of EUCLID exploits techniques from the field of user–interface design for the assessment of human–computer systems. The four components repectively call upon the specialties of Fox, Smolensky, King, and Lewis.

1. Reasoned discourse and EUCLID

1.1. The goal: Enhancing reasoned discourse

A central activity in theoretical research is the construction of reasoned arguments supporting theoretical conclusions. The problem we address is a practical one: how can this activity be effectively supported? While our focus is on argumentation of the type found in research papers (for we intend to develop our tool while using it ourselves in our work), we also consider, to a lesser extent, related forms of argumentation. Other examples of *reasoned discourse*, in addition to research papers, include policy advocacy for decision making (e.g. reasoned letters to the editor), pedagogy in theoretical disciplines such as linguistics and physics, and, to a certain extent, reasoned argumentation in everyday conversation.

The domains of analysis we have in mind are ones that are not strictly formal, so that mathematically rigorous proofs are not possible. Our goal is to enhance reasoned discourse that now occurs in natural—not formal—language.

We take the goal of enhancing reasoned discourse to integrally incorporate both support of explicit discourse processes (like reading and writing) and also support of reasoning itself. A clear distinction between cognition and communication is particularly problematic in the area of reasoning. Argumentation is the construction of a symbolic structure intended to persuade through conformity to certain social conventions: reasoning is intrinsically a discourse phenomenon. The point is underlined by a substantial body of research on writing which suggests that what writers most need support for is the *planning* of documents: the main problem is deciding exactly what to say, and devising an overall presentation plan (Flower & Hayes, 1980; Gregg & Steinberg, 1980; Kellogg, 1985a, 1985b, in press). Even in the writing of few-paragraph business letters, people spend two-thirds of their time planning (Gould, 1980). In the domain of reasoned discourse, it is clear that planning—laying out the line of argumentation—is a crucial and almost completely unsupported activity. The process of planning a research paper—the working out of the claims to be made and the arguments to support them—is essentially the process of carrying out the theoretical component of the research itself.

Our goal, then, is to provide a tool to facilitate reasoning and enhance reasoned discourse, particularly writing.

1.2. The proposed tool

In this section we specify the functionality that ARL and EUCLID are intended to provide. Methods for realizing these systems will be discussed in sections 4.2 and 4.3, respectively, after we have argued for the value of the proposed functionality and have considered related research.

1.2.1. ARL: An Argumentation Representation Language

Our fundamental hypothesis is that in constructing an argument, two kinds of knowledge are brought to bear: knowledge of the subject domain, and knowledge of argumentation per se. These respectively manifest themselves as argument content and argument structure. A *general purpose* argumentation tool helps the user by virtue of its knowledge of argument structure, not argument content.

Drawing a clear line between structure and content is so crucial to this research that we find it useful to give that line a concise name: *the Divide*. Content information is *below* the Divide; structure information is *above* the Divide. Examples of assertions below the Divide are:

- Lower interest rates lead to bull markets.
- Linguistic principle X is universal.
- Approach Y to knowledge representation is seriously flawed.

Above the Divide we find statements such as:

- Claim C_1 supports claim C_2.
- Claim C is the main point of argument A.
- Claim C is made by author S.
- Claim C_1 made by author S_1 contradicts claim C_2 made by author S_2.
- Term T is used by author S_1 to mean phrase D_1 but by author S_2 to mean phrase D_2.

This latter sort of information is often not explicitly stated in text, but in it lies the structure that characterizes reasoned discourse. (The crucial importance of information above the Divide has also been emphasized by Zukerman and Pearl (1985). In the tutoring context, they have studied how such information is introduced through natural language expressions they call *meta-technical utterances*.)

Information below the Divide involves terms and predicates that vary completely from one domain of argumentation to another. But information above the Divide involves a reasonably constant vocabulary: the examples above use the terms *claim C*, *argument A*, *author S* and the predicates *supports, main-point, asserts, contradicts*. This vocabulary is characteristic of reasoned discourse in any domain. (In section 4.2 we address the issue of variations in *styles* of argumentation across fields.)

ARL is intended to offer a set of primitive term-types and primitive predicates for formally describing argument structure, such as those mentioned in the previous paragraph. In addition, it should incorporate high-order structures formally defined by combining simpler ones. Examples that will be explicitly discussed in section 4.2 include high-level standard schematic structures for arguments, arguments by analogy, allegations of misrepresentations, and argument refutations.

To give users the expressive power needed in real argumentation, ARL must let users extend the language's set of primitives and must provide the machinery for them to formally create their own high-order constructs.

The ARL statement corresponding to "Claim C_1 supports claim C_2" uses the formal predicate **supports** to relate two entities that have formal type **claim**. The content of each claim is not expressed formally, but *informally,* in natural language. For example, the content of C_1 might be "Lower interest rates lead to bull markets." Thus ARL is a *semi-formal* language: argument *structure* information (above the Divide) is represented *formally*, while argument *content* (below the Divide) is represented *informally*. The computer has access to the semantics of the formal information, but only the user has access to the semantics of the informal information.

1.2.2. EUCLID: An Environment for User Construction of Logical Informal Discourse

A formal representation of the structure of the argument contained in a theoretical research paper is too large for anyone to explicitly represent without the help of a data manager. The computer environment EUCLID[1] must keep track of ARL representations, allowing users to select portions of the argument to be displayed on a high-resolution graphics terminal. In addition to displaying ARL structures, EUCLID must allow users to add new structures, and modify old structures. Users should be able to state that they want to create a new instance of a higher-level construct (say an analogy), whereupon EUCLID would prompt the user for the necessary inputs and manage the details of creating the necessary data structures. Information must be displayed by procedures specially designed to show the various argument components. For example, there must be special procedures for displaying analogies, refutations, or retrieval requests like "show all claims whose validity depend on this one." Along with the capability to create new types of argument structures, users need the capability to specify new procedures for displaying them. Of course users should have considerable opportunity for choice among alternative display methods.

An argument created in EUCLID would contain full pieces of text: EUCLID is intended to provide a unified environment for working out an argument *and* expressing it in text. A specific example will be illustrated in the next section; to give the general idea, reading a "journal article" in EUCLID might proceed like this. After reading the abstract, the user would decide what further information is of most interest: an experimental procedure, a source for a "fact," a theoretical argument, the theoretical assumptions. The requested information would be retrieved and displayed according to the type of information. The retrieved information might be a piece of text, like a section of a paper, or perhaps a table or graph. In one unified ARL datastructure is contained both the underlying logical structure and the pieces of text that present the argument. In this sense, EUCLID is a *hypertext system* specially tailored for logical material—reasoned discourse.

1.2.3. An example: The Chinese room debate

As an example of how EUCLID might look to a user reading an argument, we will consider an argument that has been our testbed: the "Chinese room" argument of John Searle (1980). This argument claims to show that instantiating an AI program—even one that could answer questions indistinguishably from a human and thereby pass the Turing test—cannot be sufficient grounds for saying that a machine "understands" in the full sense of the word. The core of the argument is the following analogy. A Chinese story is slipped under the door of a closed room, and then Chinese questions about the story are slipped in. Back under the door come Chinese answers to the questions, indistinguishable from those of a native speaker. As it happens, inside the room is Searle himself, working away at copying Chinese symbols he doesn't understand from big books under the guidance of a complex set of English instructions. According to Searle's analogy, the Chinese characters are to the Searle in the room as English is to a question answering computer: completely meaningless forms being manipulated without any understanding.

The commentary from numerous cognitive scientists that was published with the article revealed a tremendous diversity of outlooks, and appeared to evidence a considerable amount of confusion about just what Searle's argument was. The argument is still highly active today, meriting an entire session of the 1986 meeting of the Society of Philosophy and Psychology. Our goal is to use EUCLID to delineate, as clearly as possible, the positions taken by the numerous participants in the published debate; in the process, the expressive adequacy of ARL and the usability of EUCLID will be challenged by a truly worthy argument.

Figures 2 through 7 illustrate how EUCLID might be used to study an ARL representation of the Chinese room debate. We imagine the user has read the text, and is ready for an analysis. Figure 2 gives a tabular display of the top level of Searle's argument. This is a relatively clean display, in which a lot of relational information is implicit in the arrangement of items on the screen. Figure 3 shows the relationships explicitly. The left side of the diagram are claims and arguments that Searle attributes to his opponents, those accepting the position of "strong AI." On the right side of the diagram are the claims and arguments that Searle accepts. At the very top of the left side is the main claim of the strong AI view. Immediately beneath the strong AI position are three arguments supporting it, which Searle attributes to his opponents; to the right of each one is Searle's counter-argument. Below these counter-arguments are Searle's arguments in favor of his position, the main claim of which is stated at the top of the right column. To the left of Searle's arguments are refutations of them which them he attributes to his opponents and to the right of these are his counter-arguments.

A user facing the austere display of Figure 2 might request that all implicit relationships be made explicit, giving rise to Figure 3. Next we suppose the user to have selected "The Chinese Room" for further information about Searle's key argument. As in all hypertext systems, this selection leads to a new display, Figure 4, which expands upon the selected item. (Note that the new display is coherently displayed without intervention by the user.) Figure 4 shows the form of the Chinese room argument: it is an analogy, and is displayed in an appropriate form. On the left side of the Chinese Room display are the elements of the Chinese room domain; on the right side are the corresponding elements of the AI system. Searle's analogy is a mapping that carries elements and claims from the Chinese Room domain into elements and claims of the AI domain.

Having been shown the explicit form of the Chinese Room analogy, we next suppose that the user wishes to consult the text to check the accuracy of the analysis given for the analogy. Figure 5 shows the text separated into pieces that are explicitly connected to components of the analogy analysis. Deciding this representation of the relationships is too messy, the user requests a simpler representation. Figure 6 show the text, unbroken, next to the analogy analysis. After selecting a particular item in the analysis, the corresponding parts of the text become underlined.

After studying the analysis side-by-side with the text, the user decides to accept the analysis for the time being and proceed. A request to remove the text and the explicit relationships gives the relatively clean screen of Figure 7, from which it is now reasonable to proceed by selecting another part of the overall argument for analysis.

1.2.4. Creating arguments

Having described the kinds of capabilities EUCLID is intended to provide for reading arguments, we now consider argument generation. It is useful to distinguish a number of processes which must all be supported by EUCLID. These processes are closely related to processes that have been studied in the creation of ordinary text (Flower, Hayes, Carey, Schriver, Stratman, to appear; Hayes & Flower, 1980). The processes are intermingled, and should not be viewed as serial phases.

- *Dump:* Generate terms and assertions the author feels to be central to the argument. EUCLID serves as electronic paper.
- *Reader-preprocess:* Indicate for various terms and assertions what they assume about the reader: background, interests, what other items have been previously read (**prerequisite** relations), etc. EUCLID stores this information for use in the linearization process (below).
- *Organize:* Insert definitions of terms, relations between assertions (eg., **supports**, **contradicts**). EUCLID serves as ARL structure editor and browser.
- *Fill–in:* Generate missing terms, claims, and arguments. EUCLID provides templates for common arguments types, checks for missing components of these argument templates, and satisfies useful database queries (e.g. "find claims lacking **supports** links"). EUCLID's library of examples of different argument types lets the user browse for possible approaches.
- *Linearize:* Impose on parts of the argument graph a linear order, thereby generating a document. EUCLID partly automates this process, making use of **prerequisite** relations, and filtering the database on intended readers' background and interests. Schemas for document types help guide this process.
- *Edit prose:* Generate readable text. EUCLID functions as text editor integrated into ARL structure editor.

The support provided by EUCLID in these processes is substantial. While the activity of expressing an argument in ARL form is itself helpful in developing the argument, the support that EUCLID can supply on the basis of the formal ARL relations that it can process is a major part of the benefit of using EUCLID instead of pencil and paper for developing arguments. On the other hand, we feel that the process of expressing an argument in ARL form requires extensive human processing and that the prospects for automating the process are dim.

2. Effectiveness of EUCLID

In this section we argue that a system with the capabilities that EUCLID is intended to offer can significantly enhance human reasoning capability. By offering abstraction tools, EUCLID can assist users in comprehending and constructing arguments. By considering the example of algebraic notation, we innumerate three powerful techniques by which a language can extend cognitive abilities. We then show how EUCLID incorporates those techniques.

2.1. Techniques for overcoming cognitive limitations

Arithmetic word problems that we cannot solve intuitively "in our heads" can often be readily solved by the use of algebraic notation. How does this notation overcome our cognitive limitations?

To address this issue we adopt a simplified picture of human information processing which is crude but sufficiently accurate for our purposes. The value of such simple cognitive models in the analysis of human–computer interaction has been advocated (and demonstrated) by a number of researchers; see, for example, Card, Moran, & Newell (1983), and Norman & Draper (1986).

A critical cognitive limitation is the small capacity of working memory. Since cognitive operations can be performed only on the contents of working memory, it is crucial that relevant operands be in working memory at the time of each operation. Problems that introduce enough information to exceed the very limited capacity of human working memory thus pose serious difficulties.

One technique afforded by algebraic notation is simply the *elimination of irrelevant information* from attention, thereby freeing up working memory capacity. The algebraic representation of a word problem has discarded much information in the problem that is irrelevant to its solution: whether x represents "the age of Peter's father" or "the number of nickels" is not expressed by the notation, and when solving the equations we can free our minds of these distractions.

A second technique algebraic notation uses is *explicit chunking*. New variables can be introduced for combinations of others, and parentheses can also be used to isolate subexpressions. Then the entire chunk can be manipulated as a single entity, taking up much less working memory capacity than if the constituents were individually represented. For example, we can see that the next operation to perform is to take the square root of an entire subexpression in parentheses, while being temporarily oblivious to the contents of that subexpression.

A third technique algebra affords is *explicit decomposition*. Parentheses can be used to group together elements into subexpressions. Individual equations also form explicitly delimited subparts of the entire problem representation. Attention can be focussed on individual limited components and the rest of the problem representation can be temporarily ignored. If the subexpressions or equations that are created are sufficiently small, they can individually be contained in working memory and operated upon. Once the component has been processed, attention can return to the context in which it is imbedded. The notation keeps in *external memory* the relation between the component and the context, so we know how to incorporate the new result into the temporarily forgotten context.

The techniques of explicit chunking and explicit decomposition are in fact two sides of the same coin. Explicit chunking allows us to ignore what goes on *inside* a substructure, while explicit decomposition permits us to ignore what goes on *outside* a substructure. Each of these aspects is so powerful that they merit separate consideration.

In the next section we will argue that EUCLID can apply these three techniques to the domain of argumentation. While individual practitioners of reasoned discourse no doubt have various personal devices, explicit and implicit, for achieving the ends of these three techniques, we believe there is considerable advantage in developing a *standard, explicit notational system* for representing arguments. Before the widespread dissemination of algebraic notation, it was no doubt true that experts with "word problems" had personal methods for solving them: but consider the fact that today nearly any child can solve such problems. This is because of the powerful social advantages of using a standardized explicit representation:

- Since the same standard approach to problem representation and solution is used over and over, procedures for representation and solution can be *automatized*. Thus after explicitly using parentheses for chunking for a long time, we learn to *see* chunks without explicitly marking them with parentheses. This represents a truly significant expansion of cognitive ability.
- A standard explicit notation enhances information retrieval.
- A standard explicit notation permits explicit, systematic *instruction*.
- An explicit representational system can be *studied*. It can be perfected for power, consistency, and naturalness.
- A standard representational system can be *publically shared*. This means it can evolve under the benefit of multiple users, and, most of all, that a common language is provided as a basis for communication.

It may be too much to hope that a system like EUCLID, after the benefit of decades of development and cultural evolution, would make it possible for school children to assess the soundness of an argument with an effectiveness that is limited today to expert reasoners. Nonetheless, we believe there is ample reason to believe that resources expended in this project will be well repaid.

2.2. EUCLID: sources of power

The three techniques of the previous section are all exploited by EUCLID, as we now argue. (Our arguments are rather similar to those given for the domain-independent decision support systems Goddess and Convince developed by Judea Pearl and collaborators: Kim, 1984; Pearl, Leal, & Saleh, 1982. In EUCLID, however, the user performs a greater portion of the analysis, so our arguments are more crucial.) We begin by considering EUCLID's support of argument comprehension.

2.2.1. Applications to argument comprehension

The formal relations of ARL represent the structure of an argument and ignore its content. For many aspects of argumentation, of course, content is essential, but for many other aspects, details of content are irrelevant. EUCLID exploits the *elimination of irrelevant information* technique by letting users focus attention on structural issues and ignore content, when this is appropriate. When an argument is recognized as an *ad hominem* attack, for example, the content of the attack is usually irrelevant: what is important is just the structural fact that what is being attacked is the *source* of a claim and not the claim itself. *Once an argument has been expressed in ARL,* there are many important tasks that can proceed without reference to content:

- determination of whether a certain claim is supported or assumed;
- determination of whether a group of claims are alleged to be consistent;
- finding all claims which become unsupported if a given claim is denied;
- finding pivotal claims on which a large fraction of the conclusions depend.

In fact all of these could be defined as formal operations which the computer could carry out automatically. These operations correspond to those that can be performed on algebraic expressions without attention to what the variables denote.

Perhaps the single most difficult aspect of assessing an argument is finding components of the argument to focus on that are small enough to be understood and evaluated yet large enough to be self-contained. The argument structuring operations of ARL allow argument creators to help the reader with this task. The author can clearly mark the main claims as such, mark which claims are used to support each of these, and which claims are used to support each of these, and so on down to the unsupported assumptions at the base of the argument. This permits readers to assess the argument in small pieces. A reader adopting a top-down strategy would pick one of the main claims and ask EUCLID to present the first layer of claims on which it is based. The author should have structured the argument so that, after a reasonable period of attention, this first layer of claims can all be held simultaneously in working memory. The reader can then assess whether the alleged conclusion does indeed follow from the premises offered. If so, the reader can proceed to assess each of the premises in exactly the same manner. A reader preferring a bottom-up strategy would start with the unsupported claims (the assumptions), and examine each for acceptability. This reader would then ask EUCLID to supply the layer of claims that the author has supported solely by the assumptions, and would examine each to see if it is indeed a justified conclusion. In either case, whenever an inference or claim is deemed unacceptable by the reader, EUCLID can automatically display which parts of the argument then become invalidated. If the reader is uncertain about a claim, EUCLID can automatically display the parts of the argument to which this uncertainty propogates. (Ultimately EUCLID will be extended to support *numerical* assignment and propopogation of uncertainty.) Thus the technique of *explicit decomposition* is at the center of EUCLID's power.

One of the central mechanisms of ARL is the encapsulation of parts of the overall argument into discrete subarguments, and the encapsulation of the conclusions of each subargument into discrete claims. Thus the technique of *explicit chunking* is heavily used.

There is an important reason to believe that with practice EUCLID can provide readers of reasoned discourse with substantially improved comprehension compared to ordinary text. Research in text comprehension has shown how important it is for understanding and retention for readers to construct mental *macrostructures* in which the contents of the text are integrated. What EUCLID provides is a way for authors to make macrostructures *explicitly available* to readers rather than relying, as in ordinary text, on many subtle and often ambiguous cues for implicitly communicating macrostructural information. Of course, we do not mean that EUCLID will remove all possibilities for multiple interpretations of a text; it will merely narrow the range of interpretations, to the extent pursued by the author. The advantages cited in the previous section for having standard, publically shared, explicit notations rather than implicit, idiosyncratic representations apply with considerable force to macrostructures.

In a related vein, it can be noted that by supporting those processes of document creation described in section 1.2.4, EUCLID provides some of the help that research into the cognitive aspects of writing has shown that writers need: support for the intricate planning needed to satisfy the many constraints implicit in the difficult task of writing

(Collins & Gentner, 1980; Flower & Hayes, 1980).

2.2.2. Applications to argument construction

The three techniques discussed in section 2.1 offer considerable power during the construction of arguments as well as during their comprehension. As in ordinary writing, the author of a EUCLID argument spends considerable time wording sentences. However, with EUCLID, the author's real work has just *begun* when all the claims are framed. Now the author must explicitly represent the relationships between the claims using the formal relations of ARL. Here is where the power of EUCLID comes into play.

Explicitly representing the structure of arguments (e.g. by diagramming) appears to be quite helpful to students of informal logic (Govier, 1985). Explicitly indicating that a certain claim allegedly follows from a definite set of other claims defines a separate component of the entire argument which can be examined in isolation. Focussing one's attention completely on this component often reveals an inadequacy in the inference, and forces reformulation of the claims or abandonment of the argument component. This is an example of the usefulness of explicit decomposition in argument construction.

In the process of explicitly indicating the structural relations in an argument, the following will occur rather often. In looking to attach a claim to the conclusion that it supports, no explicit claim will be found. This is because it is common in ordinary text for a "point" (or conclusion) of a passage to be implicit in the entire passage, never being explicitly stated in a single sentence. The result will be the creation of a new claim that explictly states a conclusion of the argument in the passage. This is one way EUCLID encourages explicit chunking: the new claim encapsulates many others.

Of course, during the construction of an argument an author must continually assess the current state of the argument, so that the advantages EUCLID offers argument comprehension are also advantages for argument construction. For example, the fact that EUCLID can display those claims currently not explictly supported by other claims can be used by an author to check whether the set of unsupported claims is the intended set of assumptions or whether there remain claims that were intended to be supported but are not. Here the author benefits from the ARL formalization of argument *structure* that results from the elimination of irrelevant (*content*) information.

3. Relation of EUCLID to existing systems

There is a large amount of research related to the EUCLID project. In this section we discuss the considerable previous work that EUCLID can benefit from. We also consider a number of existing projects with goals that have been frequently confused with those of EUCLID. Certain serious (but illusory) obstacles tend to appear as a result of these confusions.

3.1. Projects requiring formalization of domain knowledge

There are a number of research projects related to EUCLID that involve formalization of domain knowledge. It is useful to contrast these with EUCLID, in which domain information is represented *informally:* it is below the Divide.

Unlike AI projects in text comprehension and generation, the EUCLID project does not involve a computer program that "understands" natural language statements about a domain. Such understanding requires formalizing domain knowledge. In EUCLID, the user, not the computer, deals with natural language statements about the domain. Thus is avoided the necessity to specify massive quantities of background knowledge about the domain of argumentation, which would be lethal to EUCLID.

Several existing AI projects in text comprehension deal specifically with argumentation. The *argument molecules* of Birnbaum, Flowers and McGuire (Birnbaum, 1982; Birnbaum, Flowers, & McGuire, 1980; Flowers, McGuire, & Birnbaum, 1982) and the *argument units* of Alvarado, Dyer, and Flowers (Alvarado, Dyer, & Flowers, 1985) are important examples of higher-level constructs that are definable in ARL. In EUCLID, however, these structures are used to help people build and comprehend arguments—not to get AI programs to understand them.

Some of the most impressive research in philosophy, formal semantics, and AI addresses the formalization of human reasoning. Formalization of reasoning about uncertainty and counterfactuals has led these researchers to various forms of nonmonotonic logic, such as circumscription and default logic (eg., *Artificial Intelligence*, 1980) to fuzzy logic (eg., Zadeh, 1979, 1984), to logics with modal operators (eg., Hintikka, 1969) and to other sophisticated forms of logic. The semantic problems lead to complex issues of model theory (eg., Asher, Bonevac, Kamp, & Smith, 1983). The point here is that these subtleties have arisen from attempts to *formally* characterize the *validity* of realistic arguments. In EUCLID, validity is not assessed by the computer but by the human. Again, assertions about the domain, upon which validity rests, are below the Divide, and their content is not formalized in EUCLID. Since no attempt is made to formally assess the validity of arguments, EUCLID avoids the subtleties and complexities that that would entail.

As an example of the alternative strategy, Bennett (1984a, 1984b, 1985; Isard & Lewis, 1984) is applying sophisticated formalizations of reasoning to computer assessment of the validity of arguments in political policy. Such a project is considerably more ambitious than EUCLID, although, for reasons argued in the previous section, we feel that EUCLID could significantly contribute to argument assessment in areas such as policy analysis.

Like Bennett's project, the OWL (McGee, 1986) is an attempt to formalize the *content* of arguments. It differs from Bennett in that the logic used is the propositional calculus: arguments must be coded in terms that avoid the complexities often found in real arguments. These simplifications make formal assessment of the validity of arguments infeasible in the OWL, which, like EUCLID, leaves assessment up to the user. The formal internal structure of claims—basically "if *p* then *q*"—is used only for information retrieval: a search for *p* produces a list of those claims containing *p*, separated according to whether *p* is in the antecendent or consequent role. We are encouraged by the successful application of the OWL to policy issues of realistic scale.

While the methods of the OWL seem appropriate for the applications to which it has been applied, we feel a more elaborate structure is needed to represent general arguments. The OWL analysis of arguments distinguishes "claims" from "evidence": each claim can have a piece of evidence supporting it (such as a reference or a "fact"). By contrast, in EUCLID, argument structure is recursive: an argument is a network of claims which are supported not by atomic entities called "evidence" but rather by potentially highly complex entities which are themselves arguments. Thus the OWL does not address itself to the issue of argument structure in the sense of EUCLID. The OWL approach seems appropriate to a special class of arguments that involve shallow reasoning from a broad base of data.

3.2. Projects not formalizing domain knowledge

There are a number of research areas that do not involve formalization of domain information and on which EUCLID can build.

Normative theory of informal logic is readily available in the philosophical literature on argumentation (for example, Acock, 1985; Engel, 1980; Fogelin, 1982; Govier, 1985). The argument strategies and fallacies identified in this literature provide higher-order argument structures, some of them quite complex, that should be formalizable in ARL; they have played a large role in the development of ARL to date, and will continue to do so. (Specific examples are discussed in the Appendix.)

Formal proof techniques (Solow, 1982) are also suggestive in this regard, for standard procedures like proof by contradiction, proof by induction, and the like have their counterparts in informal reasoning. It will be interesting to see how well EUCLID can support the proof technique of adapting an existing proof to a similar problem. We are also interested to determine the extent to which another highly stylized form of argumentation, legal reasoning, can provide us with useful insights. There are a number of techniques common to legal argumentation and, say, that of Searle (such as analogy and emotional coloration); at the same time, the strong dependence of American legal reasoning on particular cases gives it a rather distinctive style.

A pair of projects at Xerox Palo Alto Research Center have goals that are quite close to those of EUCLID. The first is NoteCards™ (Brown & Newman, 1985), which gives the user computerized files of notecards. The user can create types of links between cards and use them for cross-referencing. When a new notecard is selected, it appears on the bit-mapped screen, to be given a convenient location by the user.

Like EUCLID, NoteCards allows the user to interrelate pieces of text. It was, at least in part, conceived as a system to support argumentation. However, it is not specialized to the argumentation domain, and it is up to the user to work out a personal argument representation system. Also, the user must continually manage the screen to keep it usable. At least for a user who is an AI researcher, the system has proved to be most valuable in strengthening arguments (VanLehn, 1985).

EUCLID can be thought of as a kind of specialization of NoteCards to the argumentation domain. A standard set of argument-structuring relations are provided, which the user can extend if necessary. The standard argument structures are recognized by EUCLID as having certain natural display forms, so that the management of the screen is handled automatically in a way driven by the semantics of the specialized domain. For the user who wishes to extend the argument-structuring relations, tools are provided for specifying appropriate displays for the new relations.

Another Xerox project, IdeaSketch (Brown & Newman, 1985), provides tools for structuring pieces of text. Here there is a conceptually infinite two-dimensional electronic blackboard allowing pieces of text to be written and related items to be connected by lines. EUCLID is intended to offer views of arguments as such graphs (see Figure 4); EUCLID graphs, however, are constrained by the structure of the argumentation domain, and contain links with labels whose semantics are known to the computer. The graphical representation is only one of those available in EUCLID, and one good only for displaying certain kinds of information formally represented in ARL (binary relations such as **supports**).

Since EUCLID constitutes an extension of research directions that have been pursued at Xerox, we are establishing research relationships with PARC that will hopefully permit us to build quite directly upon their previous work.

Another related project is Notepad©, developed at the University of California, San Diego by Allen Cypher. This system offers tools like those of NoteCards for managing interrelated pieces of data, but also provides support for managing interrelated *activities*. Like NoteCards, Notepad is a very general tool, and EUCLID can be viewed as specializing the functionality of Notepad to the domain of reasoned discourse. The EUCLID user is offered support for activities such as information retrieval and editing that is driven by the semantics of the argumentation domain.

A popularly used piece of software that is relevant is the ThinkTank™ system for constructing outlines. Like EUCLID, it aims to help users sort out their ideas and to prepare for writing a well-organized piece of text. However, for argumentation, ThinkTank is simultaneously too general and too constrained a tool. Outlines are basically trees with a single type of link. This single link, if taken to cover all the relations in an argument, is much too general to offer crucial distinctions between relations in argumentation. If the single link is taken to represent a specific argument relationship, then the system allows consideration of only one relation at a time, and the imposition of tree structure is an inappropriate constraint.

Finally, the Hypertext idea championed by Ted Nelson (1974) also shares an inspiration with EUCLID. Hypertext is an active, computerized extension of text in which users continually make selections of further material to be added to the screen. A hypertext is a collection of textual (or graphical) elements linked with explicit structural relations. (For example, structural issues in a hypertext-inspired approach to on-line documentation are analyzed in O'Malley, Smolensky, Bannon, Conway, Graham, Sokolov, & Monty, 1983, and Smolensky, Monty, & Conway, 1984.) The hypertext idea is appealing, but constraining the course of events towards relatively fruitful interactions turns out to be a serious problem. In passing from the completely static form of standard text to a completely unconstrained form of hypertext, the possibility of guidance from a capable author gets lost—and so do readers. EUCLID has the dynamic aspects of hypertext, but in a context that is well constrained by ARL and the structure of argumentation. EUCLID can be looked upon as a form of hypertext specialized to reasoned discourse.

4. Feasibility of EUCLID

4.1. Task analysis

4.1.1. Informal logical argumentation: A learned skill

It is clear from looking at argumentation in natural conversations that the sort of reasoned discourse required in academic discussions, which EUCLID is intended to promote, is a learned, non-natural skill. Argumentation in conversation does not involve step-by-step reasoning from premise to conclusion, with each step supported and justified by mutually agreed upon sources of evidence. Although participants involved in a conversation are sensitive to pressures to produce evidence to support their claims—at least in this society—most often disagreements center on claims based on personal experience or personal belief, and these claims tend to be supported by repeated reference to personal statements ("I think x") or by reference to "common" knowledge ("nobody does x" or "everyone knows y"). Often if one type of support doesn't clinch the disagreement, another, logically contradictory, type is offered. Parties in a disagreement are sometimes dismissed by argument ad hominem, where their character, rather than their arguments, is impuned. The most common types of support for claims we have found in spontaneous conversations are the following:

- citing personal experience;
- citing someone else's experience;
- citing a higher authority;
- appeal to "common" knowledge;
- simple restatement of claim;
- impuning character of interlocutor.

A close examination of academic argumentation reveals a rather different set of common strategies for supporting a claim:

- citing other people in the field (higher authority);
- citing self;
- presentation and analysis of data;
- appeal to widely accepted premises which lead deductively to claim;
- appeal to something said earlier in the article;
- providing an analogy;
- giving an example;
- support by reputation (i.e., no support for a claim need be given overtly if the writer is considered an expert on the point being made).

The strategies common in conversation tend to involve personal experience and personal involvement (cf. Pomerantz, 1984), while the strategies common in academic argumentation tend to involve abstract knowledge external to the writer's everyday experience. This finding is echoed in the cross-cultural psychology literature, particularly in the work of Scribner (1982). In her research in Liberia, Scribner found that people without formal western education solved syllogisms by invoking their knowledge about their immediate environment rather than by using abstract deductive reasoning, whereas people with even a limited amount of formal western schooling used some version of deductive reasoning to solve the problems. And even American students thoroughly grounded in western education make consistent mistakes in working through the implications of syllogistic premises (Johnson-Laird, 1980). Thus, for the most part people wishing to write academic articles must learn a new way of thinking about reasoning and evidence (although academic argumentation certainly involves some use of everyday strategies—even impuning the character of the "opposition").

Furthermore, the overall structure of academic argumentation is quite different from that of conversational disagreements and must be learned, much as any style of writing must be learned (see Rubin, 1980; Akinasso, 1982; Nystrand, 1982; van Dijk and Kintsch, 1983; Fox, 1984; and McHouh, 1982 discuss differences between writing and speaking in general). We know from previous studies (e.g. Fox, to appear; Ochs, 1979) that sequences in conversation tend to have relatively little hierarchical structure compared to pieces of writing, and therefore that following the structure of a conversation requires less cognitive effort than does understanding and remembering the structure of a piece of writing (van Dijk and Kintsch, 1983). When the constraints of informal logical argumentation are added, the difficulty of both production and comprehension of a tightly organized, highly complex and logically coherent argument becomes enormous, far beyond the demands of natural conversation. Since this task is so profoundly difficult, even writers who utilize strategies to facilitate ease of composition and comprehension (see e.g., Meyer & Rice, 1982; Kieras, 1982 for discussion of these strategies), and readers who are skilled at learning from complex texts (Kintsch and van Dijk, 1978), often experience lack of success when dealing with informal logical argumentation.

It should be clear from this discussion that the skills needed for successful communication in the domain of informal logical argumentation are extremely difficult to develop, especially given that they are learned relatively late in life (see Fromkin & Rodman, 1978, on the advantages of learning language-related skills early in life) and are foreign to our everyday experience. If the situation were otherwise, there would be little need for a system like EUCLID.

4.1.2. Proposed research

If EUCLID is to be successful as a writing aid, it must provide clear links between principles of everyday argumentation and principles of logical argumentation, so that users find EUCLID useful and easy to learn. If EUCLID promotes strategies of writing that are unnaturally methodical, i.e., that violate basic principles of coherent English discourse, or if it encourages a mode of argumentation that is too radically removed from what users find desirable, or if it offers a mode of text-representation that users find ineffective, then it will fail as a writing aid. It is therefore critical that the semantics of ARL be oriented to the principles of everyday argumentation, of natural written expository prose, and of logical argumentation, and that the mode of representation be psychologically and linguistically insightful. Since these principles have received little study (some exceptions include Mann and Thompson, 1985; Fox, 1984; Reichman, 1981), one component of the research we are proposing entails discovering principles of these key areas. We therefore propose the following research:

1. Analyzing argumentation in conversation to understand the processes involved in everyday conversation. As we have seen, the processes of everyday argumentation differ from those of informal logic, but it is unclear exactly what happens in the former, since natural conversation has received so little attention in this regard (Reichman, 1981 looks at debates in spontaneous conversations, but her data must be taken as non-representative of what we mean here by everyday argumentation, since most of the participants in her conversations are academics, and academics are unusual in having been strongly shaped by informal logic). We propose to analyze instances of conversational disagreement and persuasion, which will be culled from a large variety of conversations already taped and transcribed for other purposes.

Since this research involves in part examining how people say how they know something (i.e., giving evidence), it will be relevant to look at the work in linguistics in the area of evidentiality (Chafe and Nichols, 1986), which describes grammatical systems found in some languages (predominantly AmerIndian languages) for indicating how the speaker knows what is being claimed (because it is generically true, because the speaker witnessed it, because the speaker saw evidence of it, etc.). We expect that languages with such evidential systems have institutionalized patterns that are extremely common in other languages where they are handled at the discourse level rather than the grammatical level (Givón, 1979).

The techniques that will be used for analyzing these disagreements come from the tradition of Conversational Analysis (Schegloff and Sacks, 1973; Schegloff, 1976; Schenkein, 1979), which is a subarea within the framework of ethnomethodology.

2. Analyzing well-written, well-argued academic articles to understand what kinds of structures are typical of this genre. The texts that EUCLID will support should be as coherent and natural as any well-written academic article. We therefore propose to understand better the structural organization of such academic articles by analyzing a large number of academic articles, initially drawn from the disciplines of cognitive science. We will explore several methods of analysis, including an analytic technique known as Rhetorical Structure Analysis (Mann and Thompson, 1985; Fox, 1984), context space theory (Reichman, 1981), macrostructure-based analysis (van Dijk and Kintsch, 1983) and the method developed by Meyer and Rice (1982). The overall approach will be to view higher-level discourse structures as dynamically created from local structures (Fox, to appear; McClelland, Rumelhart, & the PDP Research Group, 1986; Rumelhart, McClelland, & the PDP Research Group, 1986; Rumelhart, Smolensky, McClelland, & Hinton, 1986; Smolensky, 1986, in preparation a).

By relating the analyses of everyday argumentation to those of well-argued academic discourse, we can clarify those aspects of reasoned discourse on which people can most use support, and thereby focus and direct the goals of the EUCLID system. We will use the preceding techniques to analyze texts constructed with EUCLID, compare these analyses to those of well-written articles, and use these comparisons to direct the EUCLID design.

4.2. ARL

In section 1.2.3 and Figures 2–7 we considered what a real argument might look like to a EUCLID user. Behind the various displays lies an ARL datastructure that can be presented to the user in grpahical and tabular forms but which is fundamentally represented internally in a symbolic form. In the Appendix, we discuss the underlying ARL form of argument representation: the form stored in the EUCLID database and manipulated by retrieval, editting, and display procedures. What we describe is the current conception of the language, which is still in an early stage of development.

4.2.1. Argumentation domains to be studied

To ensure the expressive adequacy of ARL, it is important to test it during development on sophisticated and complex arguments. We have been using the Searle Chinese Room debate as our testing ground, and will continue to analyze it until we have a thorough ARL representation of the entire debate. (Of course, there is no claim about the uniqueness of the analysis.) After the Chinese Room debate, we will consider actual examples of argumentation from other areas of theoretical research, such as linguistics, physics, and computer science. It is important to ensure that ARL offers the power necessary to cope with the types of argumentation found in a variety of fields. We will also use ARL in our other research to see how well it supports argument creation in realistic situations. To test the applicability of ARL to other types of reasoned discourse, we will consider arguments from everyday conversation, as discussed in section 4.1, and a sample of written arguments from a variety of areas; for example, those collected in texts such as Fogelin (1982).

ARL offers the exciting prospect of characterizing, with considerable precision, the *styles* of argumentation found in various academic disciplines and, more broadly, in various subtypes of reasoned discourse. For example, an ARL analysis of American legal reasoning would give a formal characterization of a very stylized form of case-based argumentation. We hope to make interesting contributions in this respect as a by-product of testing the generality of ARL.

4.3. EUCLID

Development of a EUCLID prototype is well underway. This prototype assumes for the most part that the database of the argument is in (virtual) memory. The next implementation will include a database management system to handle extensive databases. Figure 8 gives a block level description of one configuration of the expanded EUCLID system. The heart of the system, the object manager and the user interface, will be written in Common Lisp and is here shown running on a Symbolics Lisp machine; a Sun workstation running an object-oriented database management system is shown managing data archival.

Certain retrieval tasks in EUCLID amount to the instantiation of predicate calculus expressions containing variables. For these tasks, a Prolog-like interpreter is convenient. By implementing such an interpreter in Lisp, we can readily exploit the advantages of logic programming while retaining the powerful advantages of the unified Lisp environment.

4.3.1. Object management

The object manager will maintain the various ARL structures as objects, along with the various methods for performing analysis and constructing displays. The ARL database used by EUCLID is a heavily interconnected network of objects of many types, including terms, claims, arguments, documents, **support** relationships, and so on. An object-oriented approach is natural for providing an extensible set of data types and for defining special types of arguments. Message-passing and procedural inheritance make it easy to define display procedures for different structures that can be specially tailored where desired and inherited from more general structures otherwise. The data contained in objects will be cross-referenced, so that, for example, the object for a DOA will possess pointers to the claims it contains, and the objects for various arguments will contain pointers to those same claims.

In order for EUCLID to be a useful research tool it is important that argument structures that are carefully worked out be accessible for multiple uses. Over the past four years, an object-oriented database management system called Sembase (for Semantic database system; King, 1984; Farmer, Myers, & King, 1985) has been developed. It it written in C, runs under Unix, and supports recursively-defined objects and type/subtype (or ISA) hierarchies. Another tool, called Cactis (for Colorado ACTIve Semantics data; Hudson & King, 1986) has been developed. This system uses attributed grammars and specialized implementation techniques to support complex object structures in a fashion which allows very fast retrievals. (For an overview of object-oriented and semantic database systems, see Hull & King, 1986.)

As these tools already exist, the implementation efforts required to build a version of EUCLID which may handle large problem domains will be greatly eased. When an ARL database grows beyond that which may be stored in memory in the Symbolics system, EUCLID will use Sembase and Cactis as an external store. In order to support a multi-user environment and to allow a user to work in parallel on various pieces of a large argument structure, we will employ a demand-driven architecture in which the database component of EUCLID can process multiple requests in parallel.

An important basic operation in EUCLID is search of the database for information that pertains to a given subject. This is done both through searches of natural language strings for relevant words, and by following explicit pointers from data objects denoting topics that have be explicitly created by the user as retrieval keys. In either case, requiring exact matching between strings or keys and retrieval cues is a very restrictive constraint. A potentially powerful alternative exploits *spreading activation* to find items indirectly related to, but not exactly matching, a given search cue. This "connectionist" approach to search is directly inspired by models of human memory and offers considerable power and flexibility (Hinton & Anderson, 1981; McClelland, Rumelhart, & the PDP Research Group, 1986; Rumelhart, McClelland, & the PDP Research Group, 1986; Smolensky, in preparation a, in preparation b.) We will test the value of connectionist retrieval techniques in the EUCLID context. Preliminary investigations of connectionist retrieval in a keyword bibliographic database (Mozer, 1984) are very encouraging.

4.3.2. The user interface

To be useful EUCLID must present a large quantity of information to the user in readable formats. The figures in section 1.2.3 illustrate two information-rich display techniques to be heavily used by EUCLID: graphs and tables. Graphical representation of the **supports** relation provides a vivid display of the overall logical structure of an argument. Tabular display of particular argument types, like analogies, allows complex interrelations to be implicitly displayed by relative locations of items in the table. Other dynamic display techniques, such as highlighting elements on the screen with intensity or color, allow the display of elements possessing a property (e.g. **unsupported**) that the user wants to concentrate attention upon.

The demanding display requirements of the EUCLID system are handled by a general tool we are developing for manipulating database *perspectives* and *views*. A database perspective is a set of items currently of interest to the user. A view is a perspective together with information about how the items in that perspective are to be displayed on the screen. Changing perspective has the effect of adding or deleting items from the screen; changing views of a given perspective amounts to changing how the same information is displayed on the screen. Perspectives are being implemented as trees which specify the flow of control for producing displays. Views are implemented as particular inheritance hierarchies of display procedures for the various data types. Thus in different views, different procedures may be used for displaying the same type of object.

On input, as is customary in object-oriented systems, we use pop-up menus heavily, along with forms. If the user wishes to create an argument by analogy, for example, the definition of that subclass of the argument class allows EUCLID to prompt the user for the necessary elements, by providing a form to fill in or a menu of options to select. The system uses the ARL definition of an analogy to manage the details of creating all the constituent ARL structures for the user.

All EUCLID input/output is handled through a set of high-level graphics procedures; this *mouse-sensitive graphics package* has been specially designed for EUCLID by Brigham Bell, but is of extremely general applicability. The use of these high-level input/output graphics routines makes EUCLID highly portable: to port EUCLID to any machine running Common Lisp requires only the porting of the mouse sensitive graphics package, and, if necessary, porting the flavors system of object-oriented programming. Neither of these tasks require extensive effort, and both provide extremely general functionality that extend well beyond the EUCLID application.

4.4. Evaluation

We feel it is extremely important to monitor the effectiveness of the evolving EUCLID system by testing it with real users. We have three related but distinct objectives in this evaluation; they must be clearly separated because different methods are applicable to each. Our general framework is based on Gould and Lewis (1985).

Objective 1: Determine revisions to the EUCLID design needed to make it more useful for realistic tasks. Gould and Lewis stress the need for early attention to user tasks and requirements. While preliminary design work has been based on contact with a subpopulation of users, namely ourselves, it is clearly essential to obtain continuing feedback from a broader class of users who are not involved in the design of the system. We will follow the methodology successfully used by the NoteCards project (Brown & Newman, 1985; VanLehn, 1985): the system is used for meaningful tasks, in argument creation and analysis, by volunteer users outside the project team. Trial users will select some tasks from their own work domain, as well as performing tasks we provide. Logs will be kept of user activities, comments, and suggestions.

Objective 2: Collect detailed information about features of the EUCLID *interface that users find difficult.* While this objective is related to the broader one just discussed, getting specific feedback on interface features requires a different methodology. This specific feedback is needed to support the design iteration that good user interface design demands. We will use the "thinking aloud" technique (Ericsson & Simon, 1984), as adapted to user interface evaluation by Lewis (1982). In this approach, users are asked to make verbal comments in real time as they work with the system. These comments often pinpoint problems in the interface that are difficult to discover from performance data or post-hoc comments. We will collect thinking-aloud data on a sampled basis from the trial users discussed above.

Objective 3: Measure the effectiveness of EUCLID. Measurement of performance in complex tasks such as those supported by EUCLID is complicated by the likelihood of wide individual differences in skill and approach. Nevertheless, we will attempt to gather data bearing directly on some of the goals of EUCLID.

To assess the effectiveness of EUCLID as an argument creation and presentation tool, we will adapt methodology developed by Gould (1978a, 1978b, 1980, 1981, 1982) to study differences in communication media. Participants will be asked to prepare arguments supporting a stated conclusion; they will use EUCLID to prepare some arguments and will generate others without aid. Users will provide text representations of the arguments prepared using EUCLID, so that we will have three products to compare: arguments in text prepared without EUCLID, arguments in text prepared with EUCLID, and arguments prepared with EUCLID presented in EUCLID form. Judges will rate the effectiveness of all arguments. With appropriate balancing of treatments we can then assess how method of preparation and presentation influence rated effectiveness.

To assess the effectiveness of EUCLID as an analytical tool we will ask readers to indicate flaws in arguments presented in text form. Readers will have the use of EUCLID in examining some arguments and not others. Some of these arguments will be selected from texts on informal logic (e.g., Acock, 1985; Engel, 1980) which provide normative grounds for evaluating readers' performance. On other arguments, readers will then be asked to compare their findings with flaws listed by other readers or suggested by us, and indicate whether they accept these putative flaws as such. This will allow us to determine whether flaws accepted by most readers are more likely to be found by readers using EUCLID, providing an effectiveness measure requiring no a priori norms.

How users feel about using the EUCLID system is to some extent independent of the quality of what they produce. We will therefore directly assess the attitude of users towards the system, to see how EUCLID affects their confidence in constructing arguments.

We have so far considered the assessment of EUCLID's benefits. An adequate evaluation must also assess costs, in performance time and learning. While tradeoffs relating quantitative measures of benefits to costs are difficult to assess, we will measure time used, and training time required, in the studies just described.

APPENDIX: ARL

In this appendix we address some of the more technical aspects of the argument representation language ARL, as it stands in its present early stage of development.

ARL uses a predicate calculus representation which is rarely (if ever) seen by casual users; it is most important to developers and extenders of the language. We shall use more or less standard predicate calculus notation in this appendix; we are still exploring various alternative notations.

1. Primitives

1.1. Data types

The basic data types of ARL include:

terms
definitions
claims
arguments
documents
strengths
domains of assertion

Terms, definitions and atomic claims are primitive entities with associated strings giving their natural language expressions. Complex claims can be built up from simpler ones using functions introduced below. For example, **asserts(SEARLE, AI programming is basically trivial)** is a composite claim that Abelson makes in the Chinese room debate.

Arguments are either simple—a single claim—or complex—built up out of claims using the logical operators of predicate calculus. Thus a conjunctive argument **a** may be constructed from several claims as **a = c1 & c2 & c3**. Below we give examples of argument types that are defined by fairly complex constructions.

In addition to pure argument structures, the EUCLID database also contains pieces of text, potentially quite extensive. When using EUCLID to write an academic paper, for example, not only is an argument structure constructed, but a document is also created. Thus ARL contains the data type **document**. Most documents are complex structures comprised of ordered sequences of sub-documents: a paper is a sequence of sections, a section is a sequence of subsections, and so on. Thus documents are generally hierarchical structures built of other documents. At the bottom of this hierarchy are primitive documents which are simply pieces of text. A primitive document may be the natural language expression of the content of a term, definition, claim, or argument.

Strengths exist so users may optionally mark assertions with a strength or confidence level; they can be numbers or strings as the user prefers.

A *domain of assertion* (DOA) is a set of claims that are embraced by one participant in an argument. Often a DOA can be labelled by a person, but DOAs at other scales are also important. Thus a DOA can be associated with a whole school of thought ("strong AI"), an individual ("Skinner"), a period in an individual's career ("late Wittgenstein"), a document ("Schank's commentary on Searle's paper"), or even a portion of an argument where some hypothetical state of affairs is entertained. Because DOAs are sets of claims, they are subject to the usual set-theoretic operations and relations (intersection, inclusion, etc.).

1.2. The model of argumentation

Domains of assertion form the basis for our model of argumentation. Participants in an argument are each entitled to a private set of claims. The semantic constraint (not syntactically enforced in ARL) is that the claims in a domain of assertion, when semantically interpreted, should be mutually consistent. The goal of persuasion is to make another participant realize that, because of claims that are already accepted, others should be accepted too to avoid inconsistency. Unlike in formal proofs, there is no prior agreement on a finitely-specified set of axioms. At any point, a participant can arbitrarily assert "if *p* then *q*" (as though this were axiomatic) and use it to derive *q* from *p*. To be persuasive, however, the claim "if *p* then *q*" must be one that someone else will accept as it stands—otherwise, it should be justified by further argumentation.

1.3. Predicates and functions

ARL contains functions that construct composite claims from simpler constitutents. A very basic function is **asserts**; in the example above,

asserts(SEARLE, AI programming is basically trivial)

we construct the claim meaning "Searle asserts that AI programming is basically trivial" by inserting the DOA **SEARLE** and the atomic claim meaning "AI programming is basically trivial" into the function **asserts**. This is in fact a claim that Abelson makes in the Chinese room debate.

We will display the syntax of the **asserts** function as

asserts(DOA, claim; strength)

where the data type of the arguments are indicated, and the arguments following ";" are optional. (In the preceding example, there is no indication of the strength with which Searle is claimed to make his assertion.) Examples of other basic functions for constructing composite claims include:

supports(argument, claim; strength)
contradicts(claim1, claim2)
relevant-to(claim1, claim2)
requires(argument, claim)
definition-of(term, definition, DOA)

There are also functions that construct claims about documents, including:

expresses(document, argument)
topic(document, term)
summary(long-document, short-document)
paraphrase(document1, document2)

Then there are some basic ARL predicates that relate terms, claims, arguments, and documents. Examples include:

contains(super-document, sub-document)
precedes(sub-document, sub-document, super-document)
contains-term(claim, term)
conjunct-of(argument, claim)
constitutent-of(argument, claim)

2. Extensions

Users can extend the primitive data types, functions, and predicates of ARL. As a simple example, to define the argument type **confusion**, a user might create a primitive function

confuses(DOA, claim1, claim2)

meaning that the point of view of DOA has confused **claim1** with **claim2**. Or to define the argument type **argument_by_analogy**, a user might choose to create the predicates

domain-of(claim, term)
maps(analogy_map, term1, term2)

The idea here is that an analogy consists of two domains: the antecedent domain about which information is presumed known (e.g. "the Vietnam war"), and the consequent domain about which a conclusion is being argued for (e.g. "Central American involvement"). The argument consists of a set of claims about each domain, and the assertion that there is a mapping between the terms occuring in those claims that takes true statements in the antecedent domain into true statements about the consequent domain. This idea will be formalized into an ARL definition below.

3. Constructs

ARL lets users define higher-order, composite data types. This is a crucial feature of ARL, one that illustrates the true sense in which ARL provides users with a *language* with real generative power. For this reason we provide a number of examples in this section of the construction of higher-order argument types. These examples are intended to convey the flavor of ARL.

We shall consider constructions of increasing complexity. For the more complex cases, we find the expressions to be considerably more transparent when infix notations are used for the central functions **asserts** and **supports**. Thus for **asserts(SEARLE, claim)** we write **SEARLE: claim** and for **supports(argument, claim)** we write **argument → claim**. To familiarize the reader with the infix notations, we shall initially write all expressions in both the more verbose and the more compact notations. In all cases we will paraphrase the formal definitions in English. Throughout this appendix we use upper-case variables (eg., X) to denote DOAs (eg., **SEARLE**) and lower-case variables (eg., **arg,cl**) for everything else: arguments, claims, etc.

The first two constructions we illustrate would be used to argue that someone is misrepresenting another's claim. We define a **strong_misrepresentation** to exist when **X** says that **Y** asserts **cl**, but in fact **Y** asserts **–cl** (**not(cl)**). We write the definition, in the compact and verbose forms, as:

```
strong_misrepresentation( X, Y, cl) :=        strong_misrepresentation( X, Y, cl) :=
        X: Y: cl  &                                   asserts( X, asserts ( Y, cl))  &
        Y: –cl                                        asserts( Y, not( cl))
```

For example, in analyzing the Chinese room debate, the following claim might be generated:

SMOLENSKY: strong_misrepresentation(ABELSON, SEARLE,
writing a program to pass the Turing test is basically trivial)

Here Smolensky asserts that Abelson has strongly misrepresented Searle. The function **strong_misrepresentation** constructs from its inputs an argument that consists of the conjunction of the claim that "Abelson says that Searle says **c**" and the claim that "Searle says **not–c**," where the meaning of **c** is "writing a program to pass the Turing test is basically trivial."

A related type of argument is:

```
weak_misrepresentation(X, Y, cl) :=
        X: Y: cl   &
        –Y: cl
```

```
weak_misrepresentation( X, Y, cl) :=
        asserts( X, asserts ( Y, cl))  &
        not( asserts( Y, cl))
```

This argument claims that "X says that Y asserts **cl** but in fact Y does not assert **cl**."

Another rather simple construction is:

```
omission( X, arg, claim1, cl2) :=
        X: arg → cl1  &
        requires( arg, cl2)  &
        –justified( X, cl2)
```

```
omission( X, arg, cl1, cl2) :=
        asserts(X, supports( arg, cl1)  &
        requires( arg, cl2)  &
        not( justified( X, cl2))
```

Here, the argument is: X has appealed to the argument **arg** to support the claim **cl1**, but **arg** actually requires for its validity that claim **cl2** hold, and X has not justified **cl2**. In other words, there is an omission in X's argument. (**justified(X, claim)** can be defined basically as "there exists an argument in the DOA X that supports **claim**." That an argument rests on an unstated claim cannot be derived from other ARL relationships, so **requires** is a primitive function.)

An example of a rather more involved construction is:

```
confusion( X, arg, cl1, cl2) :=
        X: arg  &
        X: arg → cl1  &
        arg  &
        arg → cl2  &
        confused( X, cl1, cl2)
```

```
confusion( X, arg, cl1, cl2) :=
        asserts( X, arg)  &
        asserts( X, supports ( arg, cl1)) &
        arg  &
        supports ( arg, cl2)  &
        confused( X, cl1, cl2)
```

Here the argument is that X has mistakenly concluded **cl1** from **arg** instead of **cl2**, because X has confused **cl1** with **cl2**. Searle employs this argument, claiming that Strong AI has confused "observer-relative ascriptions" of understanding with "intrinsic" understanding.

The following more elaborate structure defines when one argument, **ref**, **refutes** another argument, **arg**:

```
refutes( ref, arg) :=
        arg = claim cl1 &
                (E cl2) (contradicts( cl1, cl2) & ref → cl2)
        or
        arg = conjunction( a[ ] ) &
                (E i) refutes( ref, a[i])
        or
        arg = disjunction( a[ ] ) &
                (i) refutes( ref, a[i])
```

Here we use an informal notation in which **arg = claim cl1** means "the argument **arg** is simply a claim, **cl1**"; **arg = conjunction(a[ ])** means "the argument **arg** is the conjunction of the sub-arguments **a[i]**, **i** = 1, ..., N"; **(E i)** means "there exists an **i** in 1, ..., N"; and (i) means "for all **i** in 1, ..., N".

Two important observations should be made about this definition. First, it shows the value of recursion in ARL. Second, using this definition, a user can ask to instantiate an argument **ref** to refute **arg**, and get significant support: EUCLID can inspect the ARL definition of **arg** to see if it is a simple claim, a conjuctive argument, or a disjunctive argument, and then use the definition to guide the user in constructing the refutation. In this sense, the definition of **refutes** offers a procedural semantics for the ARL primitives **conjunction**, **disjunction** and **contradicts**.

Our next example is a rather involved definition of an argument by analogy. Searle's Chinese room analogy is displayed in tabular form in Figure 7, and some of the relationships are graphically displayed in Figure 4. The idea formalized in the following definition (along the lines of Gentner, 1983) is that there is an antecedent domain (**dom[1]**) and a consequent domain (**dom[2]**), a set of objects in the antecedent domain (**obj[1][j]**), a corresponding set of objects in the consequent domain (**obj[2][j]**), sets of corresponding properties or predicates (**pred[1][k]**, **pred[2][k]**) and claims (**cl[i][n]**) about the two domains that use the specified objects and predicates. The conclusion of the argument by analogy is a claim about the consequent domain that corresponds to a certain claim—which we call the **clinch**—about the antecedent domain. The correspondences are set up by a mapping **m**, and the argument assumes that all the claim made about the domains, except the conclusion, are true. Here is a function that constructs an analogy **a**, a mapping **m**, and uses them to create an **argument_by_analogy**:

```
argument_by_analogy( a, conclusion, dom[ ], obj[ ][ ], pred[ ][ ], cl[ ][ ], clinch) :=

        a :=    (i,j) domain-of( obj[i][j], dom[i])  &
                (i,k) domain-of( pred[i][k], dom[i])  &
                (i,n) domain-of( cl[i][n], dom[i])  &
                (j) maps( m, obj[1][j], obj[2][j])  &
                (k) maps( m, pred[1][k], pred[2][k])  &
                (n) maps( m, cl[1][n], cl[2][n])  &
                maps( m, clinch, conclusion)  &
                (i,n) cl[i][n]  &
                clinch
        &
        a → conclusion
```

The **maps** statements are to be interpreted as follows. The mapping of objects and predicates from **dom[1]** to **dom[2]** are unconstrained stipulations that define **m**. Since the various **cl[i][n]** involve these objects and predicates, the assertion that **m** maps **cl[1][n]** onto **cl[2][n]** is semantically constrained to be consistent with the stipulations defining how **m** maps objects and predicates between the domains. Assessing this consistency involves the content of the claims, i.e. information below the Divide, and must be done by the user. This is part of the evaluation of the validity of any argument by analogy. As always, the ARL definition formally characterizes the *structure* of an argument by analogy, but the assessment of the *validity* of such an argument goes beyond the formal component of ARL.

The complex relations defined by **argument_by_analogy** can be conveniently displayed in tabular form, as illustrated in Figure 7. The tabular display is defined with reference to the preceding definition by:

a

domains:	**domain[1]**	**domain[2]**
elements:	**object[1][j]**	**object[2][j]**
	predicate[1][k]	**predicate[2][k]**
	claim[1][n]	**claim[2][n]**
clinch:	**clinch**	**[conclusion]**

Here it is implied that the entries for various values of **j,k,n** are to be stacked vertically.

As our final example, we consider a higher-order construct that can serve as a template for the top level of an argument. In fact, it is the top level of Searle's argument, and was shown in Figures 2 and 3. This type of argument might be called **refute_and_support**. The idea is that an opposing point of view is credited with a claim, and a series of arguments supporting this claim are presented and refuted. Then a contradictory claim is put forth and arguments supporting it are offered. Possible counterarguments may then be formulated and refuted.

An ARL definition for this argument schema, written in a chatty, second-person style, is:

```
refute_and_support( YOU, your_claim, my_claim, your_argument[ ],
    my_refutation[ ], my_argument[ ]; your_refutation[ ], my_counter[ ]) :=
        YOU: your_claim  &
        (i) YOU: your_argument[i] → your_claim  &
        (i) refutes( my_refutation[i], your_argument[i])  &
        contradicts( my_claim, your_claim)  &
        (j) my_argument[j] → my_claim  &
        (j) YOU: refutes( your_refutation[j], my_argument[j])  &
        (j) refutes( my_counter[j], your_refutation[j])
```

As with the argument by analogy, the complex interrelations of the components of the **refute_and_support** argument lend themselves to tabular display, as shown in Figure 2. The tabular form is related to the preceding definition by:

YOU	ME
your_claim	my_claim
your_argument[i]	my_refutation[i]
	my_argument[j]
your_refutation[j]	my_counter[j]

Here **ME** represents the DOA in which the **refute_and_support** argument is made. (In Figure 2, **your_refutation[1]**, the Strong AI refutation for Searle's "argument from formality: the Chinese room," happens to be a disjunction of three sub-arguments. Thus **my_counter[1]** is a conjunction of three sub-arguments. **my_argument[2]** in Figure 2 is Searle's "argument from lactation," for which no refutations are shown.)

We now have a glimpse of what is going on behind the scenes of Figures 2 and 3. Internally, the top level of Searle's argument is represented in ARL as a **refute_and_support** structure. Associated with this type of argument is a datastructure expressing the preceding tabular format. There is a display procedure that can use this datastructure to display intances of **refute_and_support** arguments in tabular form. This is what generates the display in Figure 2. The relations between elements in the table are all encoded in the underlying ARL definition of **refute_and_support**; these relations can be retrieved and then explicitly displayed using a procedure that draws labelled arcs. This is how Figure 3 is generated. Similar procedures using the **argument_by_analogy** definition and tabular dispay format generate Figures 7 and 4. Figures 5 and 6 involve an ARL function that was mentioned in passing in section 1.3: **expresses.** It links a claim in an ARL analysis of the Chinese room debate to the piece of text in the original document that allegedly expresses the claim. Two alternative procedures for displaying the **expresses** function are respectively illustrated in Figures 5 and 6.

References

Acock, M. (1985). *Informal logic examples and exercises.* Belmont, CA: Wadsworth.

Akinasso, N. (1982). On the difference between spoken and written language. *Language & Speech, 25,* 97–125.

Alvarado, S.J., Dyer, M.G., & Flowers, M. (1985). Memory representation and retrieval for editorial comprehension. *Proceedings of the International Joint Conference on Artificial Intelligence.*

Artificial Intelligence. (1980). Special issue on non-monotonic logic. Volume 13, Numbers 1–2.

Asher, N., Boenva, D., Kamp, H., & Smith, C. (1983). Discourse representation. Proposal to the National Science Foundation.

Bennett, J.P. (1984a). Data stories: Learning about learning from the U.S. experience in Vietnam. In S. Sylvan & S. Chan, Eds., *Foreign Policy Decision Making: Perception, Cognition, and Artificial Intelligence.* Praeger.

Bennett, J.P. (1984b). No seconds thoughts about 'no first use'—How can the policy analyst undertake subjunctive reasoning in political argument? Manuscript.

Bennett, J.P. (1985). Pre-theory for comparing national security policies. Paper prepared for New Directions in the Comparative Study of Foreign Policy, Columbus, OH. May.

Birnbaum, L. (1982). Argument molecules. *Proceedings of the American Association for Artificial Intelligence.*

Birnbaum, L., Flowers, M., & McGuire, R. (1980). Towards an AI model of argumentation. *Proceedings of the American Association for Artificial Intelligence.*

Brown, J.S., & Newman, S.E. (1985). Issues in cognitive and social ergonomics: From our house to Bauhaus. *Human–Computer Interaction, 1,* 359–391.

Card, S.K., Moran, T.P., & Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale, NJ: Erlbaum.

Chafe, W., & Nichols, J., Eds. (1986). *Evidentiality.* Norwood, NJ: Ablex.

Collins, A. & Gentner, D.A. (1980). A framework for a cognitive theory of writing. In L.W. Gregg & E.R. Steinberg (Eds.), *Cognitive processes in writing.* Hillsdale, NJ: Erlbaum.

Cypher, A. (1986). The structure of users' activities. In D.A. Norman & S.W. Draper, Eds., *User centered system design.* Hillsdale, NJ: Erlbaum.

Kim, J. H. (1984). CONVINCE: A CONVersational INference Consolidation Engine. Technical Report UCLA–CSD–84. Computer Science Department, UCLA.

Engel, S.M. (1980). *Analyzing informal fallacies.* Englewood Cliffs, NJ: Prentiss-Hall.

Ericsson, K.A., & Simon, H.A. (1984). *Protocol analysis.* Cambridge, MA: MIT Press/Bradford.

Farmer, D., Myers, D., & King, R. (1985). The semantic database constructor. *IEEE Transactions on Software Engineering, SE-11*, 583–590.

Flower, L.S., & Hayes, J.R. (1980). The dynamics of composing: Making plans and juggling constraints. In L.W. Gregg & E.R. Steinberg (Eds.), *Cognitive processes in writing.* Hillsdale, NJ: Erlbaum.

Flower, L.S., Hayes, J.R., Carey, L., Schriver, K., & Stratman, J. (to appear). Detection, diagnosis and the strategies of revision. *College Composition and Communication.*

Flowers, M., McGuire, R., & Birnbaum, L. (1982). Adversary arguments and the logic of personal attacks. In W.G. Lehnert & M.G. Ringle (Eds.), *Strategies for natural language understanding.* Hillsdale, NJ: Erlbaum.

Fogelin, R.J. (1982). *Understanding arguments: An introduction to informal logic.* New York: Harcourt, Brace, Javanovich.

Fox, B. (to appear). *Discourse structure and anaphora in written and conversational English.* Cambridge, England: Cambridge University Press.

Fromkin, V., & Rodman, R. (1978). *Introduction to language.* New York: Holt, Rinehart, & Winston.

Gentner, D.A. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science, 7*, 155–170.

Givón, T. (1979). *On understanding grammar.* New York: Academic.

Gould, J.D. (1978a). An experimental study of writing, dictating, and speaking. In J. Requin, Ed., *Attention and Performance VII.* Hillsdale, NJ: Erlbaum.

Gould, J.D. (1978b). How experts dictate. *Journal of Experimental Psychology: Human Perception and Performance, 4*, 648–661.

Gould, J.D. (1980). Experiments on composing letters: Some facts, some myths, and some observations. In L.W. Gregg & E.R. Steinberg (Eds.), *Cognitive processes in writing.* Hillsdale, NJ: Erlbaum.

Gould, J.D. (1981). Composing letters with computer-based text editors. *Human Factors, 23*, 593–606.

Gould, J.D. (1982). Writing and speaking letters and messages. *International Journal of Man-Machine Studies, 16*, 147–171.

Gould, J.D. & Lewis, C.H. (1985). Designing for usability: Key principles and what designers think. *Communications of the ACM, 28*, 300–311.

Govier, T. (1985). *A practical study of argument.* Belmont, CA: Wadsworth.

Gregg, L.W. & Steinberg, E.R., Eds. (1980). *Cognitive processes in writing.* Hillsdale, NJ: Erlbaum.

Hayes, J.R. & Flower, L.S. (1980). Identifying the organization of writing processes. In L.W. Gregg & E.R. Steinberg (Eds.), *Cognitive processes in writing.* Hillsdale, NJ: Erlbaum.

Hintikka, K.J.J. (1969). *Models for modalities.* Dordrecht: Reidel.

Hinton, G.E. & Anderson, J.A. (1981). *Parallel models of associative memory.* Hillsdale, NJ: Erlbaum.

Holt, R.C. (1983). *Concurrent Euclid, the UNIX™ system, and Turis.* Reading, MA: Addison-Wesley.

Hudson, S., & King, R. (1986). Cactis: A database system for specifying functionally-defined data. *Proceedings of the International Workshop on Object-Oriented Database Systems.*

Hull, R., & King, R. (1986). Semantic database modeling: Survey, applications, and research issues. Technical report TR-86-201. COmputer Science Department, USC.

Isard, W., & Lewis, B. (1984). James P. Bennett on subjunctive reasoning, policy analysis, and political argument. *Conflict Management and Peace Science, 8,* 71–112.

Johnson-Laird, P.N. (1983). *Mental Models.* Cambridge, MA: Harvard University Press.

Kellogg, R.T. (1985). Computer aids that writers need. *Behavior Research Methods, Instruments, & Computers, 17,* 253–258.

Kellogg, R.T. (1985). Why outlines benefit writers. Paper presented to the Psychonomic Society, Boston.

Kellogg, R.T. (in press). Designing idea processors for document compostion. *Behavior Research Methods, Instruments, & Computers.*

Kieras, D. (1982). A model of reader strategy for abstracting main ideas from simple technical prose. *Text, 2,* 47–82.

King, R. (1984). Sembase: A semantic database management system. *Proceedings of the First International Workshop on Expert Database Systems.*

Lampson, B.W., Horning, J.J., London, R.L., Mitchell J.G., & Popek, G.J. (1977). Report on the programming language Euclid. *SIGPLAN Notices, 12,* Number 2.

Lewis, C.H. (1982). Using the "thinking-aloud" method in cognitive interface design. Report RC-9265. IBM Research, Yorktown Heights, NY.

Mann, W. & Thompson, S. (1985). Relational propositions. *Proceedings of the Berkeley Linguistics Society.*

McClelland, J.L., Rumelhart, D.E., & the PDP Research Group. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume II: Psychological and biological Models.* Cambridge, MA: MIT Press/Bradford.

McGee, V.E. (1986). The OWL: Software support for a model of argumentation. *Behavior Research Methods, Instruments, & Computers, 18,* 108–117.

McHouh, A. (1982). *Telling how texts talk: Essays on reading and ethnomethodology.* New York: Routledge and Kegan Paul.

Meyer, B., & Rice, G. (1982). The interaction of reader strategies and the organization of text. *Text, 2,* 141–154.

Mozer, M.C. (1984). Inductive information retrieval using parallel distributed computation. Technical report 8406. Institute of Cognitive Science, UCSD, La Jolla, CA.

Nelson, T.H. (1974). *Dream machines.* South Bend, IN: the distributors.

Norman, D.A. & Draper, S.W. (1986). *User centered system design.* Hillsdale, NJ: Erlbaum.

Nystrand, M., Ed. (1982). *What writers know.* New York: Academic.

Ochs, E. (1979). Planned and unplanned discourse. In Givón, Ed., *Syntax and Semantics, vol. 12.* New York: Academic.

O'Malley, C., Smolensky, P., Bannon, L., Conway, E., Graham, J., Sokolov, J., & Monty, M.L. (1983). A proposal for user centered system documentation. *Proceedings of the CHI 1983 Conference on Human Factors in Computing Systems.*

Pearl, J., Leal, A., & Saleh, J. (1982). GODDESS: A goal-directed decision structuring system. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI–4,* 250-262.

Pomerantz, A. (1984). Giving a source or basis: the practice in conversation of telling *Journal of Pragmatics.*

Reichman, R. (1981a). Modeling informal debates. *Proceedings of the International Joint Conference on Artificial Intelligence.*

Reichman, R. (1981b). Plain speaking: a theory and grammar of spontaneous conversaton. Technical report 4681. Bolt, Beranek & Newman, Cambridge, MA.

Rubin, A. (1980). A theoretical taxonomy of the differences between oral and written language. In R. Spiro, B. Bruce, & W. Brewer, Eds., *Issues in Reading Comprehension.* Hillsdale, NJ: Erlbaum.

Rumelhart, D.E., McClelland, J.L., & the PDP Research Group. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume I: Foundations.* Cambridge, MA: MIT Press/Bradford.

Rumelhart, D.E., Smolensky, P., McClelland, J.L., & Hinton, G.E. (1986). Schemata and sequential thought processes in parallel distributed processing. In J.L. McClelland, D.E. Rumelhart, & the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume II: Psychological and Biological Models.* Cambridge, MA: MIT Press/Bradford Books.

Schegloff, E. & Sacks, H. (1973). Opening up closings. *Semiotica, 8,* 289–327.

Schenkein, J., Ed. (1979). Studies in the organization of conversation interaction. New York: Academic.

Scribner, S. (1977). Modes of thinking and ways of speaking: culture and logic reconsidered. In P.N. Johnson-Laird & P.C. Wason, Eds., *Thinking: Readings in Cognitive Science.* Cambridge, England: Cambridge University Press.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume I: Foundations.* Cambridge, MA: MIT Press/Bradford.

Smolensky, P. (in preparation a). *Connectionist models of cognition.* Hillsdale, NJ: Erlbaum.

Smolensky, P. (in preparation b). *Principles of connectionist artificial intelligence.*

Smolensky, P., Monty, M.L., & Conway, E. (1984). Formalizing task descriptions for command specification and documentation. *Proceedings of the International Federation of Information Processing Conference on Human–Computer Interaction.*

Solow, D. (1982). *How to read and do proofs, an introduction to mathematical thought process.* New York: Wiley.

van Dijk, T., & Kintsch, W. (1983). *Strategies of discourse comprehension.* New York: Academic.

VanLehn, K. (1985). Theory reformulation caused by an argumentation tool. Report. Xerox Palo Alto Research Center, Palo Alto, CA.

Zadeh, L.A. (1979). A theory of approximate reasoning. In J.E. Hayes, D. Michie, & L.I. Mikulich, Eds., *Machine Intelligence 9.* New York: Wiley.

Zadeh, L.A. (1984). Syllogistic reasoning in fuzzy logic and its application to reasoning with dispositions. Technical Report 16. Cognitive Science Program, UCB, Berkeley, CA.

Zukerman, I. & Pearl, J. (1985) Tutorial dialogs and meta-technical utterances. Manuscript. Computer Science Department, UCLA.

Acknowledgements

This research has been supported by a grant from Symbolics, Inc., and by the Department of Computer Science and Institute of Cognitive Science at the University of Colorado at Boulder.

Figure 1: Top level of the argument offered in this report.

Numbers coincide with sections of the report.

1. *Definition:* The goal: enhancing reasoned discourse
   Definition: ARL and EUCLID (functional specifications)
   Example: Using EUCLID to read the Chinese room argument of John Searle.

2. *Claim:* EUCLID can contribute significantly to achieving the goal.
   Argument summary: Analysis of sources of the power of notation to overcome cognitive limitations suggest that EUCLID should be effective.

3 *Claim:* A system meeting EUCLID's specifications does not yet exist, but previous research provides much to build upon.
Argument summary: Consideration of several existing projects show they provide valuable foundations, but that considerable further research is necessary.

4 *Claim:* Development of EUCLID is feasible.
Argument summary: Methodologies exist for developing the necessary theoretical constructs and for effectively implementing them. These methodologies fall into several subareas, corresponding to the following particular claims.

   4.1 *Claim:* It is feasible to perform a task analysis of argumentation that reveals the aspects of the task on which people need support. This analysis fills in the details of the specifications that ARL and EUCLID are to meet.
   Argument summary: Discourse analysis provides a number of techniques for analyzing examples of reasoned discourse, and the literature on informal logic offers normative analysis of argumentation.

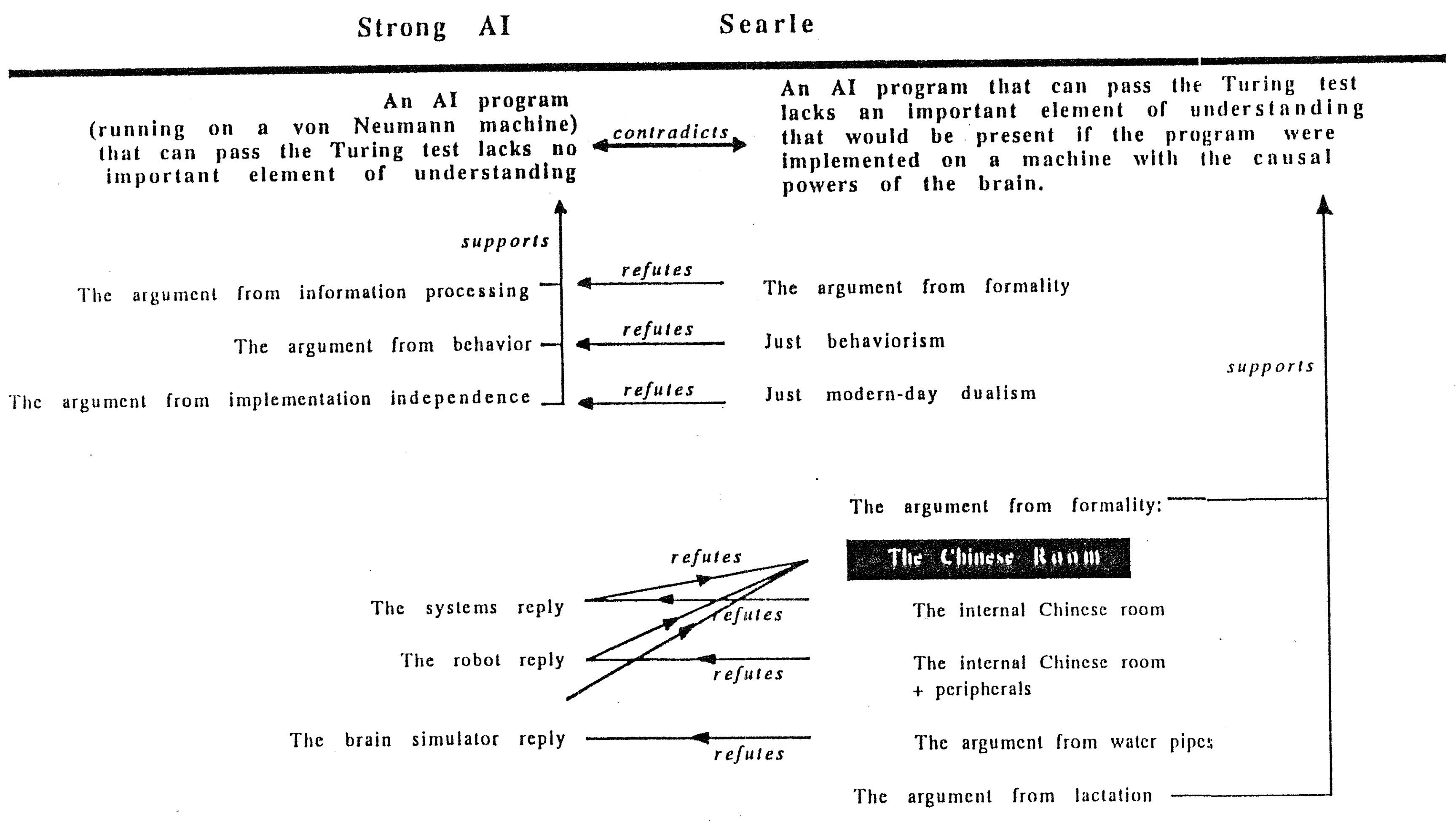
   4.2 *Claim:* It is feasible to develop a language meeting the specifications of ARL.
   Argument summary: Predicate calculus techniques can be used to develop ARL through testing on actual large-scale academic arguments. (See appendix.)

   4.3 *Claim:* It is feasible to develop a computer environment meeting the specifications of EUCLID.
   Argument summary: Object-oriented data representation, semantic data modeling, and graphical interface techniques can be used.

   4.4 *Claim:* The effectiveness of EUCLID can be assessed.
   Argument summary: Empirical techniques like those employed in human–computer interface evaluation can be used.

Strong AI	Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding	An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine with the causal powers of the brain.
The argument from information processing	The argument from formality
The argument from behavior	Just behaviorism
The argument from implementation independence	Just modern-day dualism
	The argument from formality: The Chinese room
The systems reply	The internal Chinese room
The robot reply	The internal Chinese room + peripherals
The brain simulator reply	The argument from water pipes
	The argument from lactation

Figure 2



Strong AI
Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding
contradicts
An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine with the causal powers of the brain.
supports
The argument from information processing
refutes
The argument from formality
The argument from behavior
refutes
Just behaviorism
The argument from implementation independence
refutes
Just modern-day dualism
supports
The argument from formality:
The Chinese Room
refutes
The systems reply
refutes
The internal Chinese room
The robot reply
refutes
The internal Chinese room + peripherals
The brain simulator reply
refutes
The argument from water pipes
The argument from lactation

Figure 3

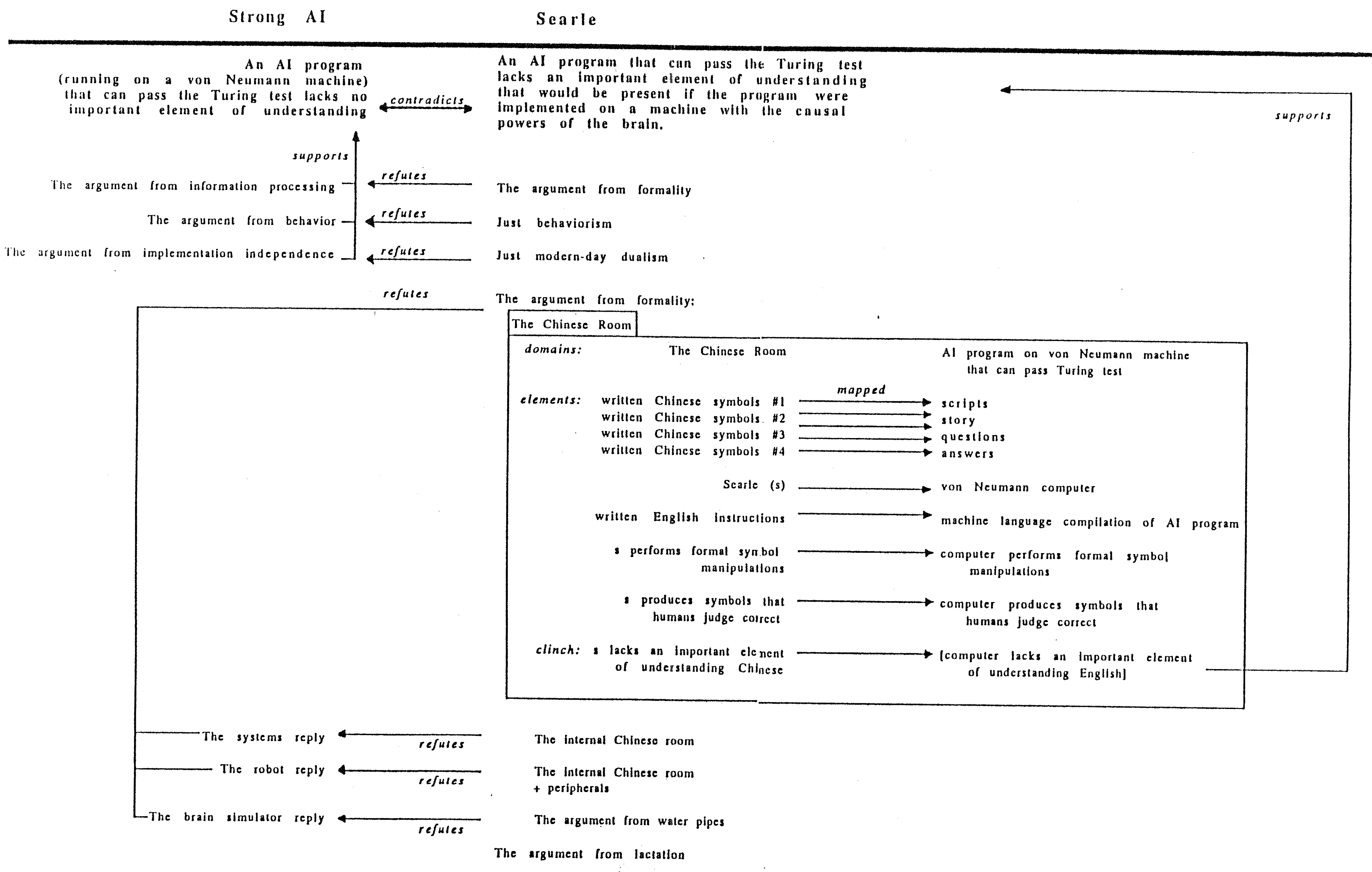
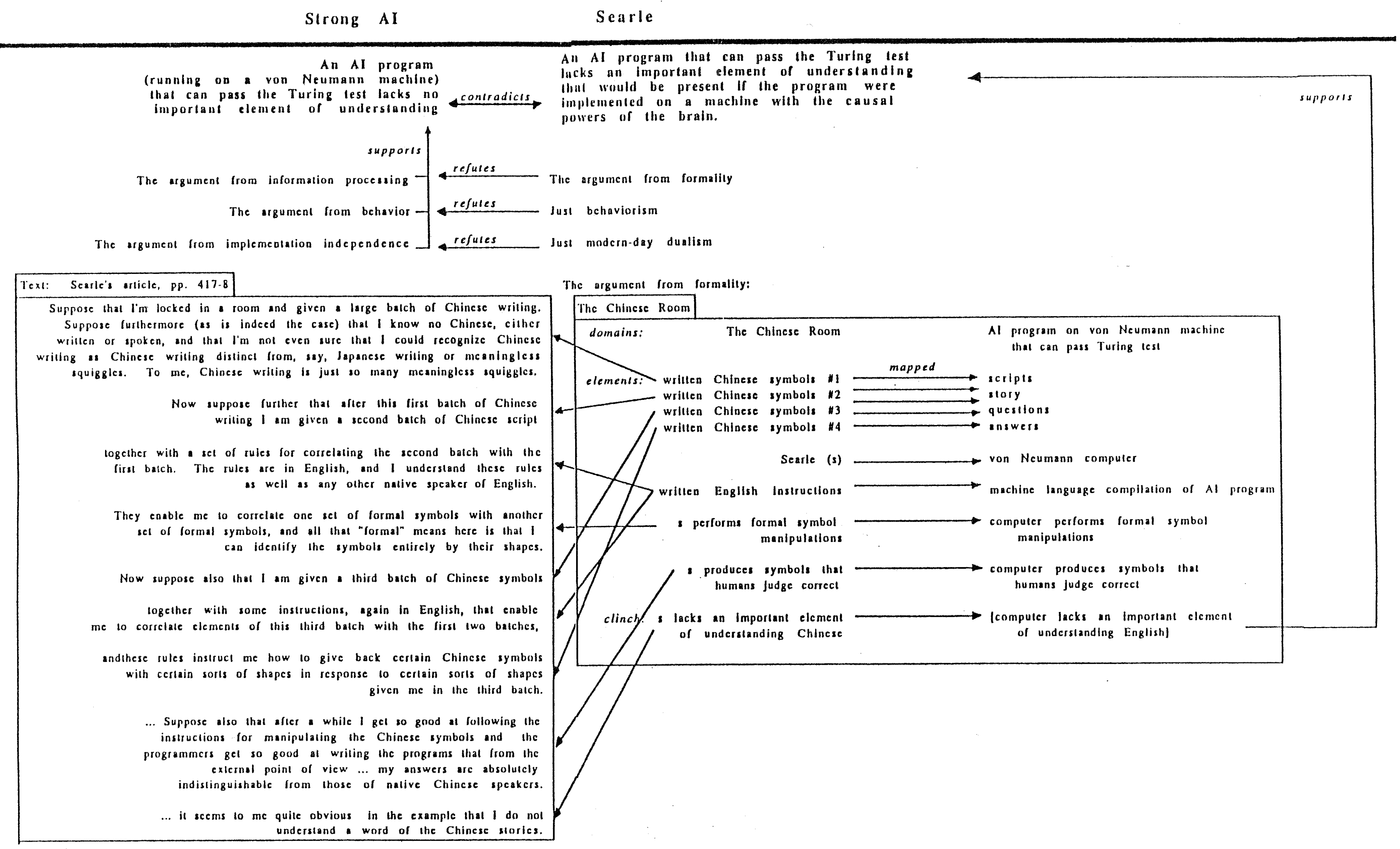

Strong AI
Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding
contradicts
An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine with the causal powers of the brain.
supports
supports
The argument from information processing
refutes
The argument from formality
The argument from behavior
refutes
Just behaviorism
The argument from implementation independence
refutes
Just modern-day dualism
refutes
The argument from formality:
The Chinese Room
domains:
The Chinese Room
AI program on von Neumann machine that can pass Turing test
elements:
mapped
written Chinese symbols #1
scripts
written Chinese symbols #2
story
written Chinese symbols #3
questions
written Chinese symbols #4
answers
Searle (s)
von Neumann computer
written English instructions
machine language compilation of AI program
s performs formal symbol manipulations
computer performs formal symbol manipulations
s produces symbols that humans judge correct
computer produces symbols that humans judge correct
clinch:
s lacks an important element of understanding Chinese
[computer lacks an important element of understanding English]
The systems reply
refutes
The internal Chinese room
The robot reply
refutes
The internal Chinese room + peripherals
The brain simulator reply
refutes
The argument from water pipes
The argument from lactation

Figure 4


Strong AI
Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding
contradicts
An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine with the causal powers of the brain.
supports
supports
The argument from information processing
The argument from behavior
The argument from implementation independence
refutes
refutes
refutes
The argument from formality
Just behaviorism
Just modern-day dualism
The argument from formality:
The Chinese Room
domains:
The Chinese Room
AI program on von Neumann machine that can pass Turing test
elements:
written Chinese symbols #1
written Chinese symbols #2
written Chinese symbols #3
written Chinese symbols #4
mapped
scripts
story
questions
answers
Searle (s)
von Neumann computer
written English instructions
machine language compilation of AI program
s performs formal symbol manipulations
computer performs formal symbol manipulations
s produces symbols that humans judge correct
computer produces symbols that humans judge correct
clinch:
s lacks an important element of understanding Chinese
[computer lacks an important element of understanding English]
Text: Searle's article, pp. 417-8

Suppose that I'm locked in a room and given a large batch of Chinese writing. Suppose furthermore (as is indeed the case) that I know no Chinese, either written or spoken, and that I'm not even sure that I could recognize Chinese writing as Chinese writing distinct from, say, Japanese writing or meaningless squiggles. To me, Chinese writing is just so many meaningless squiggles.

Now suppose further that after this first batch of Chinese writing I am given a second batch of Chinese script

together with a set of rules for correlating the second batch with the first batch. The rules are in English, and I understand these rules as well as any other native speaker of English.

They enable me to correlate one set of formal symbols with another set of formal symbols, and all that "formal" means here is that I can identify the symbols entirely by their shapes.

Now suppose also that I am given a third batch of Chinese symbols

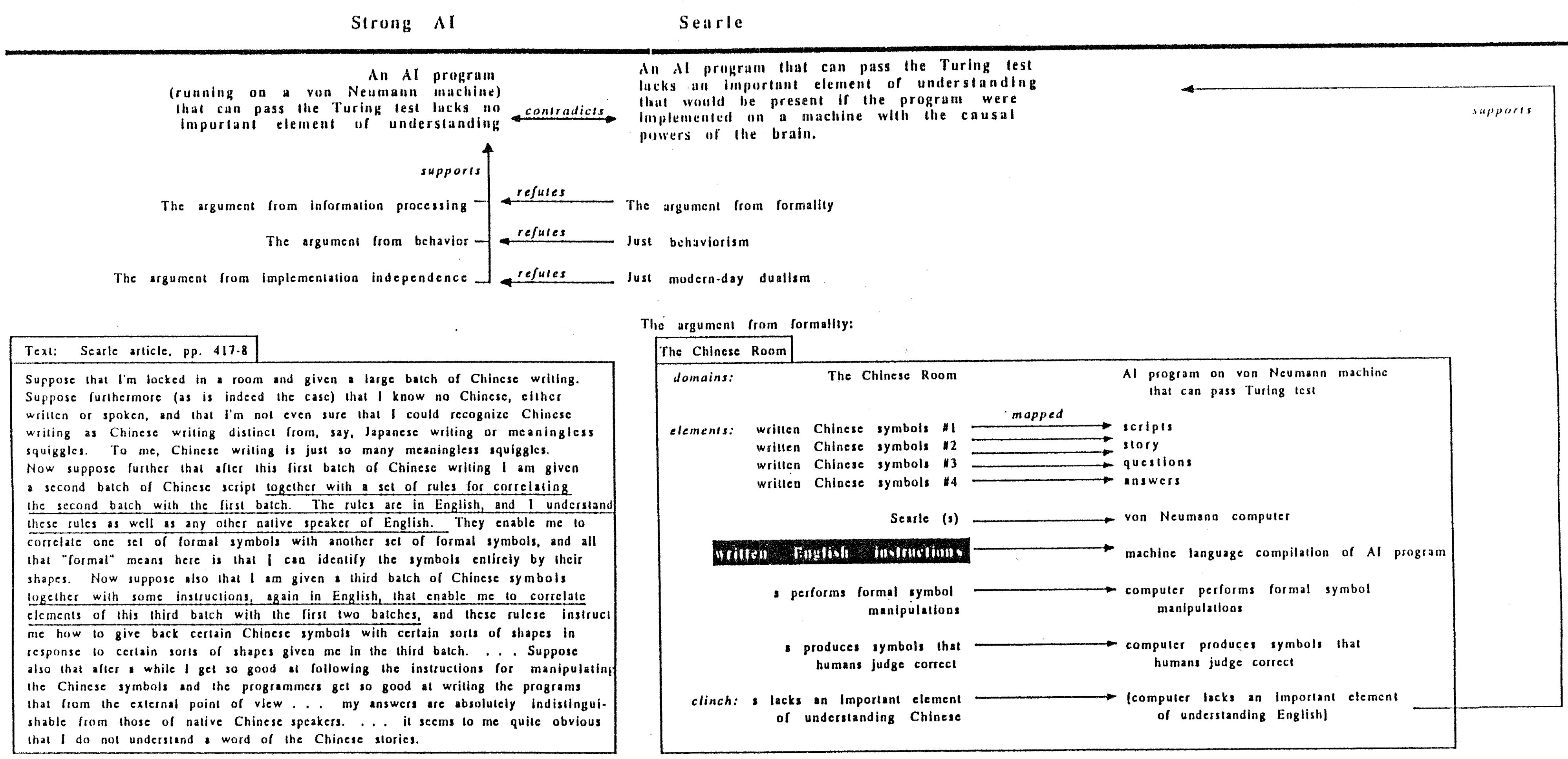
together with some instructions, again in English, that enable me to correlate elements of this third batch with the first two batches,

andthese rules instruct me how to give back certain Chinese symbols with certain sorts of shapes in response to certain sorts of shapes given me in the third batch.

... Suppose also that after a while I get so good at following the instructions for manipulating the Chinese symbols and the programmers get so good at writing the programs that from the external point of view ... my answers are absolutely indistinguishable from those of native Chinese speakers.

... it seems to me quite obvious in the example that I do not understand a word of the Chinese stories.

Figure 5


Strong AI
Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding
contradicts
An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine with the causal powers of the brain.
supports
supports
The argument from information processing
refutes
The argument from formality
The argument from behavior
refutes
Just behaviorism
The argument from implementation independence
refutes
Just modern-day dualism
The argument from formality:
The Chinese Room
domains:
The Chinese Room
AI program on von Neumann machine that can pass Turing test
elements:
written Chinese symbols #1
written Chinese symbols #2
written Chinese symbols #3
written Chinese symbols #4
mapped
scripts
story
questions
answers
Searle (s)
von Neumann computer
written English instructions
machine language compilation of AI program
s performs formal symbol manipulations
computer performs formal symbol manipulations
s produces symbols that humans judge correct
computer produces symbols that humans judge correct
clinch:
s lacks an important element of understanding Chinese
[computer lacks an important element of understanding English]
Text: Searle article, pp. 417-8

Suppose that I'm locked in a room and given a large batch of Chinese writing. Suppose furthermore (as is indeed the case) that I know no Chinese, either written or spoken, and that I'm not even sure that I could recognize Chinese writing as Chinese writing distinct from, say, Japanese writing or meaningless squiggles. To me, Chinese writing is just so many meaningless squiggles.
Now suppose further that after this first batch of Chinese writing I am given a second batch of Chinese script together with a set of rules for correlating the second batch with the first batch. The rules are in English, and I understand these rules as well as any other native speaker of English. They enable me to correlate one set of formal symbols with another set of formal symbols, and all that "formal" means here is that I can identify the symbols entirely by their shapes. Now suppose also that I am given a third batch of Chinese symbols together with some instructions, again in English, that enable me to correlate elements of this third batch with the first two batches, and these rulese instruct me how to give back certain Chinese symbols with certain sorts of shapes in response to certain sorts of shapes given me in the third batch. . . . Suppose also that after a while I get so good at following the instructions for manipulating the Chinese symbols and the programmers get so good at writing the programs that from the external point of view . . . my answers are absolutely indistinguishable from those of native Chinese speakers. . . . it seems to me quite obvious that I do not understand a word of the Chinese stories.

Figure 6

Strong AI	Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding	An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine with the causal powers of the brain.
The argument from information processing	The argument from formality
The argument from behavior	Just behaviorism
The argument from implementation independence	Just modern-day dualism

The argument from formality:

The Chinese Room

domains:	The Chinese Room	AI program on von Neumann machine that can pass Turing test
elements:	written Chinese symbols #1	scripts
	written Chinese symbols #2	story
	written Chinese symbols #3	questions
	written Chinese symbols #4	answers
	Searle (s)	von Neumann computer
	written English instructions	machine language compilation of AI program
	s performs formal symbol manipulations	computer performs formal symbol manipulations
	s produces symbols that humans judge correct	computer produces symbols that humans judge correct
clinch:	s lacks an important element of understanding Chinese	[computer lacks an important element of understanding English]

Strong AI	Searle
The systems reply	The internal Chinese room
The robot reply	The internal Chinese room + peripherals
The brain simulator reply	The argument from water pipes
	The argument from lactation

Figure 7

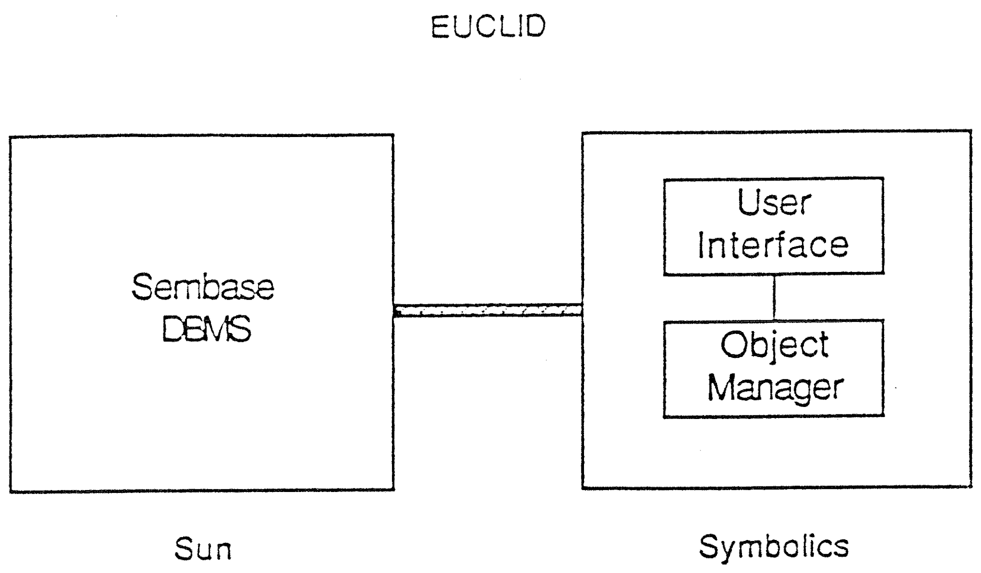

EUCLID
Sembase DBMS
User Interface
Object Manager
Sun
Symbolics

Figure 8