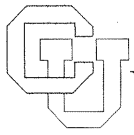
Constraint–Based Hypertext for Argumentation

Paul Smolensky
Brigham Bell
Barbara Fox
Roger King
Clayton Lewis

CU-CS-379-87



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Table of Contents

Constraint-Based Hypertext for Argumentation

Paul Smolensky[1,3] Brigham Bell[1] Barbara Fox[2,3] Roger King[1] Clayton Lewis[1,3]

[1]Department of Computer Science, [2]Department of Linguistics & [3]Institute of Cognitive Science
University of Colorado, Boulder, CO 80309

ABSTRACT

In this paper we describe a hypertext system we are developing for the support of reasoned argumentation: the EUCLID project. We use the project to address two general problems arising with hypertext: the problems of controlling user/document interaction, and the problem of controlling the screen. We suggest that guiding users' interaction with hypertext is difficult because of the unique form of discourse that hypertext represents, and that structuring user/document interaction can be achieved through specializing to a particular type of material and designing the hypertext system to respect the particular discourse structure characteristic of that material. EUCLID's design is tuned to the structure of **reasoned discourse**. *The problem of screen management in EUCLID is a serious one, because our presentation of complex arguments requires mapping the complex logical relations between parts of realistic arguments onto complex spatial relations between items in the display. We describe a general system we are developing which provides this high degree of control for hypertext screen management. This system represents a* **constraint-based** *approach to hypertext, in which the items from the underlying database that are to be displayed may each contribute a number of constraints on the layout; a general constraint-satisfier then computes a screen layout that simultaneously satisfies these constraints. Each time an item is to be added to or deleted from the screen, the constraint set is adjusted and the screen layout is recomputed; thus the spatial relationships on the screen provide at all times a veridical representation of the underlying relations between displayed database items. This kind of strong screen control is demanded by hypertext applications which, like ours, are* **fine grained**: *the number of nodes and links being displayed number in the hundreds.*

0. INTRODUCTION

This paper addresses two general problems arising in hypertext systems: the problem of controlling the interaction between user and document, and the problem of controlling the screen.

Hypertext is a new form of communication that introduces a new challenge in controlling interaction between document and user. This challenge arises because hypertext falls in a difficult no-man's-land between the traditional discourse media of ordinary text and conversation. In ordinary text, the reader has virtually no influence on the presentation of information, so the author has complete control over the organization of material, and an obligation to design an organization that will be satisfactory for the reader. In ordinary conversation, the interlocutor has great influence on the flow of information, but the speaker is physically present to make real-time decisions about how to dynamically structure that information. In

hypertext, the reader has tremendous control over the flow of information, but the author is not there to provide real-time guidance. There is thus a tendency for users of hypertext systems to be denied the kind of guidance possible in ordinary text or conversation, guidance that is generally necessary for a user to effectively absorb information from a source whose structure is complex and not thoroughly familiar. In short, hypertext systems often set up an interaction between user and document that is poorly structured, and users simply get lost.

One approach to structuring hypertext interaction is to focus on a certain type of material, and base the hypertext interaction on a study of the discourse structure characteristic of that type of material. In the work reported here, we have focussed on material that is characterized by a high degree of *logical structure*: reasoned argumentation. By focussing on this *reasoned discourse*, it is possible to develop a hypertext system designed to support user-computer interaction that revolves specifically around logical structure. Section 1 of this paper reports on the hypertext system we are developing to support reasoned discourse: the EUCLID system.[1] A fuller description of this system may be found in [Smol88].

The second general hypertext problem we address is that of controlling the screen. In presenting large complex arguments, it is important to design a well-organized screen in which many logical relationships are implicitly encoded in the spatial relations between items on the screen. This will be elaborated in some detail below, but to take a simple example, it is often useful to place opposing arguments side-by-side, using a representation in which the logical opposition of the arguments is implicitly encoded in their relative spatial locations. Such implicit notations are often desirable; they are absolutely necessary in situations where explicitly indicating all relationships (eg., by labelled arrows) would create a display that is hopelessly cluttered.

The demands of creating the highly structured displays needed to portray complex arguments has led us to an approach to screen management we call *constraint-based display*. The basic idea is that every item and relationship to be displayed contributes a certain number of constraints on the screen layout; the display system then designs an appropriate display by simultaneously satisfying all these constraints. We are developing a general constraint-based hypertext system for displaying items selected from a large network database, and applying this general system specifically to our argumentation domain.

Section 1 of this paper is a description of the EUCLID system for supporting reasoned discourse. This motivates the constraint-based approach to hypertext, and gives a concrete arena in which to then discuss the constraint-based approach in Section 2. The project is in its early phases: much of the system to be described has already been implemented; the remainder has been designed and is currently being built.

---

1. The environment EUCLID is unrelated to the programming language Euclid [Lamp77, Holt83].

1. REASONED DISCOURSE AND THE EUCLID PROJECT

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 Section of the paper, 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.

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 are developing 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 [Flow80, Greg80, Kell85a, Kell85b, Kell88]. Even in the writing of few-paragraph business letters, people spend two-thirds of their time planning [Goul80]. 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 tool

In this section we specify the functionality that ARL and EUCLID are intended ultimately to provide. Presently, only part of this functionality has been implemented.

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 [Kuck85]. 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.

ARL offers 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 incorporates high-order structures formally defined by combining simpler ones: for example, 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.

We believe that the notion of semi-formal language is potentially of great value to the design of effective joint human/computer systems: its applicability extends beyond joint human/computer reasoning, to include nearly any joint human/computer activity.

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 keeps 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 allows users to add new structures, and modify old structures. Users can state that they want to create a new instance of a higher-level construct (say an analogy), whereupon EUCLID prompts the user for the necessary inputs and manages the details of creating the necessary data structures. Information is displayed in ways specially designed to show the various argument components. For example, there are special displays for analogies, refutations, or retrieval requests like "show all claims whose validity depends on this one." Along with the capability to create new types of argument structures, users have the capability to specify new ways of displaying them. Users also have considerable choice among alternative types of displays.

An argument created in EUCLID may 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 decides what further information is of most interest: an experimental

procedure, a source for a "fact," a theoretical argument, the theoretical assumptions. The requested information is retrieved and the screen is reconfigured to incorporate the new information, which is displayed in a manner reflecting its information type. The retrieved information might be a piece of text, like a section of a paper, or perhaps a table or graph, or even a running program. In one unified ARL datastructure is contained both the underlying logical structure and the pieces of text that present the argument. 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 [Sear80]. 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 reveals a tremendous diversity of outlooks, and appears to evidence a considerable amount of confusion about just what Searle's argument is. 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 1 through 6 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 1 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 2 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 own 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 1 might request that the implicit relationships be made explicit, giving rise to Figure 2. Next we suppose the user to have selected "The Chinese Room" for further

information about Searle's key argument. This selection leads to a new display, Figure 3, which expands upon the selected item. (Note that the new display is coherently displayed without intervention by the user.) Figure 3 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 4 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 5 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 6, from which it is now reasonable to proceed by selecting another part of the overall argument for analysis.

The displays of Figures 1 through 6 are mockups towards which we are currently striving. The displays of Figures 7 through 10 are screens from the current EUCLID implementation on the Symbolics™ lisp machine. Figure 7 shows the top level of the argument; Figure 8 shows an expanded view in which "the argument from information processing" and "the Chinese Room argument and replies" have both been expanded in place. Figure 9 shows the screen after attention has been shifted to the Chinese Room argument, which now fills the screen; "the systems reply" and "the internal Chinese room" have now been expanded out. In Figure 10, we have returned to the top level of the argument; the part of the argument in which Searle refutes his opponents has been reduced to a fairly compact form, and the part in which he supports his own position has been expanded. All the operations that have been performed by the user are simple menu selections, indicating what information to add or delete; all screen management has been done by EUCLID.

The displays of Figures 7 through 10 are generated in a top-down, recursive fashion, in which large datastructures called *display templates* specify the overall organization of the large portion of the screen occupied by a particular argument; when imbedded arguments are displayed, control is governed by the templates for the imbedded arguments. The displays of Figures 1 through 6 demand an additional degree of screen control relative to those of the existing implementation; switching to the use of arrows to indicate relationships, as in going from Figure 1 to Figure 2, requires a degree of local control that is cumbersome for the current top-down, template-driven display system. To achieve the more local screen control required by Figures 7 through 10, we are currently re-implementing EUCLID using the constraint-based display approach to be described in Section 2.

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 [Flow88, Haye80]. 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.

1.3. Effectiveness of EUCLID

Three techniques for overcoming the limitations of human reasoning capacity are exploited by EUCLID. In discussing them, we will illustrate the techniques with examples from EUCLID but also from a familiar and extremely powerful reasoning tool: algebra.

The first technique is the *elimination of irrelevant information.* In algebra, one important reason we can solve problems is that once the problem has been cast into a set of equations, we can forget what the

variables denote, and not be distracted by that now-irrelevant information. The process of expressing an argument in EUCLID form has some of the same advantages as expressing a word problem in algebraic notation. Once an argument's *structure* has been expressed in ARL, a number of structural analyses can proceed without the distractions introduced by the content of the argument: for example, it is possible to find all claims that would become unsupported if a given claim in the argument were denied, or all the unsupported claims, or pivotal claims on which large parts of the argument hinge.

The second technique is *explicit chunking*. In algebra, a new variable can be introduced to denote a complex subexpression, and the simplification the new variable affords enables us to cope with expressions that would otherwise be too complex to manage. In EUCLID, a central tool for assessing arguments is to encapsulate large subarguments into a single conclusion, often, a conclusion that is only *implicit* in the original, non-EUCLID form of the argument. This encapsulation enables users to more effectively cope with complex arguments.

The third technique is *explicit decomposition*. In algebra, it is crucial to decompose a problem into a set of individual equations that can analyzed separately, or to decompose a complex expression into individual terms that can be analyzed separately. Only by isolating independent subparts can we cope with complex algebraic problems. Similarly, in EUCLID, a central technique is the decomposition of complex arguments into subarguments which can be constructed, understood, or assessed independently.

In addition to these particular techniques for overcoming cognitive capacity limits and enhancing reasoning, there are very general reasons why EUCLID can promote the effectiveness of reasoned discourse. Expressing an argument in ARL amounts to making notationally *explicit* the kind of logical structure that is often left *implicit* in informal argumentation. Explicit notational systems offer many advantages over implicit systems: standardized systems permit automatization of standard skills; they promote information retrieval; they enable explicit instruction, study and analysis of the notational system; and the public sharing of the system enables evolution of the system and provides a common basis for communication. Furthermore, there is good evidence that providing readers *explicitly* with the overall structure or *macrostructure* of documents significantly increases their comprehensibility and retention [vanD83].

1.4. Relation of EUCLID to existing systems

Because EUCLID is an AI-based system for argumentation and reasoning, the project is often identified with AI projects aimed at computer understanding of arguments in natural language form [Alva85, Birn82, Birn80, Flow82] and with attempts to formalize the principles of human reasoning (eg. circumscription and default logic—[Arti80]—fuzzy logic—[Zade79, Zade84]—logics with modal operators—[Hint69]—and so forth). In fact, the EUCLID project involves *neither* of these goals. The notation of the EUCLID system, ARL, is a *semi-formal* language, processed both by the computer and by users. Since only information about argument structure is formalized in EUCLID, there is no attempt to formalize domain information as is necessary for an AI system that understands arguments in natural language. Furthermore, the validity of arguments in EUCLID is assessed by the user, informally, so there is no attempt to formulate formal

principles by which validity is assessed in complex, realistic arguments.

Nonetheless, ARL is a language for expressing argument structures whose development is being driven in part by the need to provide the power necessary to express the *argument molecules* of Birnbaum, Flowers and McGuire [Birn82, Birn80, Flow82], the *argument units* of Alvarado, Dyer, and Flowers [Alva85] and the argument strategies and fallacies studied in the informal logic literature (for example, [Acoc85, Enge80, Foge82, Govi85]).

There are number of existing general hypertext-like systems that can be related to EUCLID. NoteCards™ [Brow85, VanL85], NotePad© [Cyph86], IdeaSketch™ [Brow85], and, to a much more modest degree, ThinkTank™, are all systems for developing and interrelating ideas. EUCLID can be viewed as a specialization of these systems to the particular domain of argumentation. The system is tuned to the specific demands of argumentation, from the display of information, to the editing operations, to the retrieval of information. One particular manifestation of this tuning is that the screen is continually managed to maintain an orderly display of information and the logical relations between items. The system assumes such a large burden of screen management that EUCLID has forced the development of a new hypertext displaying system, discussed in Section 2 of this paper.

1.5. Discourse analysis

As stated at the beginning of this paper, we feel that hypertext interaction needs to be structured in a way appropriate to the discourse type of the particular material in question. Our construction of the EUCLID system is therefore being guided by research into the structure of reasoned discourse, in currently existing media such as text and conversation, and in the hypertext medium as it emerges. This research is being carried out as part of the EUCLID project, and a number of techniques from discourse analysis are being applied. As explained at the beginning of the paper, hypertext involves a mixture of discourse elements from conversation and text, and thus provides fertile new ground for discourse analysis. A goal of our analysis of reasoned discourse is to maximize the effectiveness of EUCLID by providing users with the kind of support they most need in constructing, comprehending, and assessing arguments.

2. THE CONSTRAINT-BASED APPROACH TO HYPERTEXT

The EUCLID project requires a screen management system that allows the spatial layout of items on the screen to represent complex relations between dynamically selected items from the underlying database. Each time the user adds or deletes information from the screen, the screen needs to be redesigned so that the spatial layout of the new screen properly reflects the new information content. (Of course, it is important to minimize the screen reorganization so that users do not need to continually make major reorientation to the screen.)

EUCLID is an example of a hypertext application that is *fine grained*: the nodes and links being displayed on the screen at a given time number in the hundreds. Each node contains a small amount of content (on the order of 10 words); the hundreds of relations being displayed obviously cannot each be overtly displayed, eg. by an labelled arrow; most must be displayed implicitly by the spatial arrangement of the displayed

nodes. This is just how an outline, for example, displays the tree structure of its nodes: that node *x* is a child of node *y* is displayed by placing *x* below *y*, indented by one quantum of indentation; that node *z* is the next sibling of node *x* is indicated by having node *z* displayed immediately below *x* with the same amount of indentation. This sort of implicit representation of relations is used heavily in the EUCLID displays of Figures 1 through 6. (Even where relations are explicitly displayed with arrows in these Figures, the nodes are spatially arranged so that these relations are *simultaneously* implicitly represented in the layout.)

We are developing a general hypertext facility for providing the high degree of screen constrol demanded by fine grained hypertext applications such as EUCLID. This facility is called *CBH*, for *Constraint-Based Hypertext*. EUCLID is a particular application built on CBH.

CBH is used to display portions of network databases of the sort usually assumed in hypertext. At any given moment, there is a subset of the database that is currently being displayed on the screen: the *active perspective*. (There may be other inactive perspectives stored: these are database subsets that are not currently being displayed). All the items in the active perspective are entitled to influence the current display. The displayed items in the active perspective include all kinds of database items: objects, properties, and relations. CBH is designed for fine grained hypertext systems where perspectives can simultaneously include hundreds of nodes and links.

The layout of the current display is computed as follows. Each item in the active perspective influences the screen layout by contributing *constraints* on the layout; all the constraints contributed by all the items in the active perspective are assembled together into the *active constraint set.* A constraint satisfier then computes a screen layout that satisfies the constraints in the active constraint set. (These constraints may be satisfied in an approximate fashion.)

Examples of the kind of constraints that appear in the active constraint set are as follows. Consider a data object with textual content: say, a EUCLID claim `c`. This object might contribute to the active constraint set the constraint that in some rectangular region `R(c)` on the screen, the string giving the content of the claim `c` must be printed. This constraint can be written:

```
contains-text( R(c), content(c), Display-attributes(c))
```

Alternatively, in a more compact representation, this object might contribute the constraint that in some rectangular region on the screen, a generic icon designating "claim" must be drawn:

```
contains-graphic( R(c), claim-icon-3 , Display-attributes(c))
```

Next consider an item from the database that is not a data object but rather a property: say, the EUCLID attribute `unsupported`, applied to some particular claim `c`. This item might contribute to the active constraint set the constraint that the display of `c` must use a large bold font, or, alternatively, must appear in the color red:

```
size( Display-attributes(c))  = large
style( Display-attributes(c)) = bold
```

```
color( Display-attributes(c)) = red
```

Finally, consider a database item that is a relation between two objects: say, the EUCLID relation `refutes`, holding between two claims `c1` and `c2`. This relation might contribute to the active constraint set the constraint that the rectangles in which `c1` and `c2` are displayed must be horizontally aligned:

```
horizontally-aligned( R(c1), R(c2))
```

Alternatively, this relation might contribute the constraint that there must be an arrow bearing the label *refutes* from the display of `c1` to the display of `c2`:

```
joined-by-arrow( R(c1), R(c2), Path, "refutes")
```

The constraints contributed by displayed items involve a number of *display variables*, such as locations of rectangles on the screen (`R`) wherein information is to be displayed, paths of lines and arrows (`Path`), display attributes (such as color, font size and style: `Display-attributes`), and so on. The problem of screen layout is then the problem of assigning values to all these display variables so that all the constraints in the active constraint set are met.

As indicated by the examples above, alternative methods for displaying a given item correspond to different constraints. Choosing to represent a claim by its contents or by an icon, choosing to represent a property by font size or by color, choosing to represent a relation by spatial alignment or by an arrow, all amount to choosing for database items different constraints to contribute to the active constraint set. Each package of constraints that a database item can contribute to the active constraint set is called a *constraint schema*. For example, a constraint schema that has already been mentioned is `joined-by-arrow`: for any binary relation `r` in the database, the relationship `r(x,y)` can contribute the constraint

```
joined-by-arrow( R(x), R(y), Path(r), label(r))
```

Thus a particular screen is determined by two things: the active perspective—which determines *what* database items are displayed—together with a choice for each item of a particular diplay constraint schema to contribute to the active constraint set—which determines *how* each item is displayed. Since it will often be desirable to have all items of a particular type displayed in the same way, ie., according to the same constraint schema, it is important to be able to specify at the level of data *types* the choices of which constraint schemata are to be contributed by the various items in the active perspective. This can be arranged as follows.

Consider all the database items in a perspective. Each item has a data type, and we can tack each item onto the database type hierarchy by suspending it from the node for its type. This creates a tree, whose leaves include the database items in the perspective, and whose non-terminal nodes are datatypes. To any node *n* on this tree we can attach information specifying the constraint schema governing the constraints contributed by the given individual or type *n*. We call this tree a *display constraint schema hierarchy*, or for short, a *display hierarchy*. A valid display hierarchy has the property that every individual database

item in the tree either has a constraint schema directly attached to it, or inherits a constraint schema from some ancestor in the tree.

Thus to specify a display in CBH, it is necessary to specify a perspective and a display hierarchy. The CBH displayer takes the active perspective and for each item in that perspective, it finds in the display hierarchy the appropriate constraint schema. It then uses this constraint schema to generate the set of particular constraints to be contributed by that item. It assembles the constraints contributed by all the items in the perspective to form the active constraint set. It then uses its constraint satisfier to assign values to all the display variables occuring in the constraints. Once these values are assigned, the display is composed; the displayer then uses the computed values of the display variables to paint the display on the screen.

Figures 11 through 15 use a small example to summarize this discussion of how CMH works. Figure 11 gives the overall architecture of the CBH system. Figure 12 shows a small EUCLID-like screen. Figure 13 shows the relevant portion of the database; the active perspective corresponding to Figure 12 consists of all the instances in the portion of the database shown in Figure 13. Figure 14 shows the display hierarchy used to create the screen of Figure 12; all specification of constraint schemata is done at the type level, so the instances suspended from the types are not shown. The relation linking nodes in the display hierarchy to their types is called `display-constraint-schema`. Figure 15 gives part of the definitions of the constraint schemata used, using a Prolog-like notation in which the names of variables begin with upper-case letters.[2]

The preceding discussion has treated display constraints as *hard* constraints: predicate-calculus-like propositions that would presumably have a truth value of either *true* or *false*. Actually, many of the important constraints in screen layout are *soft*: they are constraints that can be met more or less well, and the goal of constraint satisfaction is to meet the total set of soft constraints as well as possible. Examples of such soft constraints include: make the location of an item as close as possible to its position in the previous display (assuming it was present in the previous display); make the font size as close as possible to 10 points (where smaller font sizes may be nessary to fit items onto a crowded screen); line up horizontally as well as possible the following items; place this item as high as possible on the screen; place these items as close as possible to each other; and so forth. Our approach to display constraints incorporates soft constraints as well as hard ones. As usual with soft constraints, numerical strengths are necessary, and the specification of these strengths is part of the job of the constraint schemata (and these strengths are, of course, user-modifiable). Each soft constraint contributes a term to a sum S that defines the total degree to which the constraints are violated. The goal of constraint satisfaction, then, is to minimize S. Our initial approaches to this numerical optimization problem include both the stochastic

2. The conventions for formally specifying constraints used in the text were modified slightly relative to Figure 15, for expository convenience. The intent of this paper is to communicate the basic idea of constraint-based hypertext; details of constraint specification will be taken up elsewhere.

technique of simulated annealing [Kirk83] and the deterministic technique of gradient descent.

It is important to note that in the hypertext setting, most of the constraint satisfaction problems that need to be solved are *incremental* problems. That is, the typical case is that a screen has already been laid out, and the user has just added an item to or deleted an item from the perspective; the set of constraints that now need to be satisfied differ very little from the set already satisfied, so the previous display provides a very good starting point from which to seek an optimum for the new *S* function.

Figure 11 includes a number of "editors" for modifying the various data structures in the CBH system. To illustrate how these work, imagine a user to have clicked on a displayed item for a menu of operations. Imagine the user selects *delete*. A second menu now appears, offering a number of different "deletions" available to the user: the selected item can be deleted from the perspective, so that it disappears from the screen but remains in the underlying database; or the underlying item can be deleted from the database; or the constraint schema for the selected item can be deleted from the current display hierarchy. These three forms of deletion call upon the functionality of the three editors: the perspective editor, the database editor, and the display hierarchy editor, respectively. (From the user's point of view, there is no need to distinguish the editors per se: there are simply a number of operations available through menus that alter the active perspective, the database, or the current display hierarchy.)

3. CONCLUSION

In this paper we have addressed the problems of controlling the interaction between users and hypertext, and of controlling the screen. We have suggested that user/document interaction be structured according to the particular discourse type characterizing the given material. We have described a hypertext system, EUCLID, for the support of reasoned argumentation. Interaction in EUCLID is being designed to reflect the structure of a particular discourse type, reasoned discourse. EUCLID rests on a semi-formal argument representation language, ARL, in which formal specification of structural information, for use by both user and machine, is combined with informal specification of content information, to be used only by the user. The EUCLID system assumes the burden of maintaining at all times a coherent, well-composed screen, in which complex structural relations among the underlying database items are represented by complex spatial relations among the displayed representations of those items. The great demands this places on control of the screen have led us to develop a general system for hypertext display in which displayed items contribute constraints on the display, and the display system composes a screen layout that satisfies the total set of constraints. The design of this constraint-based hypertext system CBH has been described; its implementation is now underway.

ACKNOWLEDGEMENTS

This research has been supported by NSF grant IST-8609599, a grant from Symbolics, Inc., and by the Department of Computer Science and Institute of Cognitive Science at the University of Colorado at Boulder.

REFERENCES

[Aboc85] Acock, M. (1985). *Informal logic examples and exercises.* Belmont, CA: Wadsworth.

[Alva85] 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.*

[Arti80] *Artificial Intelligence.* (1980). Special issue on non-monotonic logic. Volume 13, Numbers 1–2.

[Birn82] Birnbaum, L. (1982). Argument molecules. *Proceedings of the American Association for Artificial Intelligence.*

[Birn80] Birnbaum, L., Flowers, M., & McGuire, R. (1980). Towards an AI model of argumentation. *Proceedings of the American Association for Artificial Intelligence.*

[Brow85] Brown, J.S., & Newman, S.E. (1985). Issues in cognitive and social ergonomics: From our house to Bauhaus. *Human–Computer Interaction, 1,* 359–391.

[Cyph86] Cypher, A. (1986). The structure of users' activities. In D.A. Norman & S.W. Draper, Eds., *User centered system design.* Hillsdale, NJ: Erlbaum.

[Enge80] Engel, S.M. (1980). *Analyzing informal fallacies.* Englewood Cliffs, NJ: Prentiss-Hall.

[Flow80] 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.

[Flow88] 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.*

[Flow82] 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.

[Foge82] Fogelin, R.J. (1982). *Understanding arguments: An introduction to informal logic.* New York: Harcourt, Brace, Javanovich.

[Goul80] 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.

[Govi85] Govier, T. (1985). *A practical study of argument.* Belmont, CA: Wadsworth.

[Greg80] Gregg, L.W. & Steinberg, E.R., Eds. (1980). *Cognitive processes in writing.* Hillsdale, NJ: Erlbaum.

[Haye80] 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.

[Hint69] Hintikka, K.J.J. (1969). *Models for modalities.* Dordrecht: Reidel.

[Holt83] Holt, R.C. (1983). *Concurrent Euclid, the UNIX™ system, and Turis.* Reading, MA: Addison-Wesley.

[Kell85a] Kellogg, R.T. (1985). Computer aids that writers need. *Behavior Research Methods, Instruments, & Computers, 17,* 253–258.

[Kell85b] Kellogg, R.T. (1985). Why outlines benefit writers. Paper presented to the Psychonomic Society, Boston.

[Kell88] Kellogg, R.T. (in press). Designing idea processors for document compostion. *Behavior Research Methods, Instruments, & Computers.*

[Kirk83] Kirkpatrick, S., Gelatt, C.D. Jr., and Vecchi, M.P. (1983). Optimization by simulated annealing. *Science,* 220, 671-80.

[Lamp77] 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.

[Sear80] Searle, J. (1980) Minds, brains, and programs. *The Behavioral and Brain Sciences* 3, 417–457.

[Smol88] Smolensky, P., Fox, B., King, R., & Lewis, C. (in press). Computer-aided reasoned discourse, or, How to argue with a computer. In R. Guindon (Ed.), *Cognitive Science and Its Applications For Human–Computer Interaction.* Hillsdale, NJ: Erlbaum. Also available as Technical Report CU–CS–358–87. Department of Computer Science, University of Colorado at Boulder. February, 1987.

[vanD83] van Dijk, T., & Kintsch, W. (1983). *Strategies of discourse comprehension.* New York: Academic.

[VanL85] VanLehn, K. (1985). Theory reformulation caused by an argumentation tool. Report. Xerox Palo Alto Research Center, Palo Alto, CA.

[Zade79] 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.

[Zade84] 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.

[Zuke85] Zukerman, I. & Pearl, J. (1985) Tutorial dialogs and meta-technical utterances. Manuscript. Computer Science Department, UCLA.

FIGURE CAPTIONS

Figure 1. An example EUCLID screen showing the high-level argument structure in an analysis of the Chinese room debate. Figures 1 through 6 are Macintosh™ mock-ups from the system design. (Figures 7 through 10 show screens from an implementation.)

Figure 2. The screen of Figure 1 after relations that were implicitly represented by spatial relationships have been explicitly represented by labelled arrows. One node on the screen, "The Chinese Room," has been selected for expansion in place.

Figure 3. The screen of Figure 2 after expansion of the Chinese Room argument.

Figure 4. The screen of Figure 3 after a request for the original text underlying the individual elements in the analysis of the Chinese Room argument.

Figure 5. The screen of Figure 2 after a request for the text underlying a single selected element of the analysis. This display indicates the underlying text differently from the display of Figure 4.

Figure 6. The screen of Figure 2 after elimination of the arrows explicitly representing the relations of the displayed items; the relations are indicated only implicitly, in the spatial relations on the screen of the displayed items.

Figure 7. An actual screen image of the high-level argument structure of the Chinese room debate, from a Symbolics™ implementation of EUCLID. The overall argument structure is that of a **`refute-and-support-argument`**: opponents' arguments are stated and refuted, then supporting arguments are given. Here the **`refute`** part has three sub-arguments and corresponding refutations; the **`support`** part has a single sub-argument.

Figure 8. The screen from Figure 7 after the **`refute`** part of the argument has been expanded to show more detail.

Figure 9. The refutation of "The argument from information processing" is itself a **`refute-and-support-argument`**. Here it has been expanded to fill the display.

Figure 10. A return to the top-level of the whole argument, as shown in Figure 7, with the **`support`** sub-argument now expanded in place. It is an **`analogy-argument`**.

Figure 11. The overall architecture of the CBH (Constraint-Based Hypertext) system we are developing as a general-purpose hypertext system on which to build EUCLID.

Figure 12. A very simplified sample EUCLID screen to illustrate the constraint-based approach to hypertext.

Figure 13. The network database and the active perspective underlying Figure 12.

Figure 14. The display hierarchy underlying Figure 12.

Figure 15. Examples of definitions of some of the display constraint schemata used in Figure 14.

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 1.

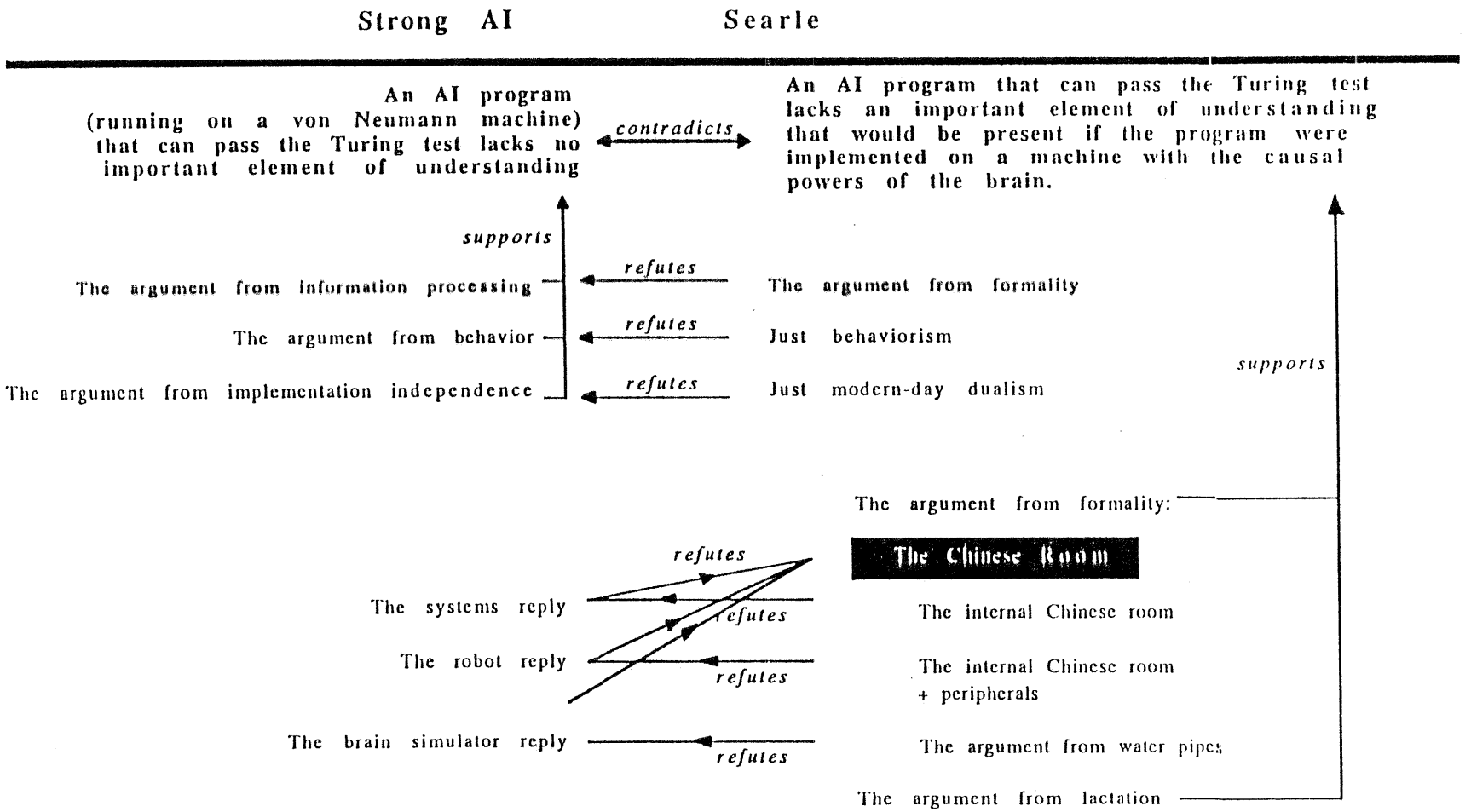

Strong AI
Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding
contradicts
An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine 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 2.

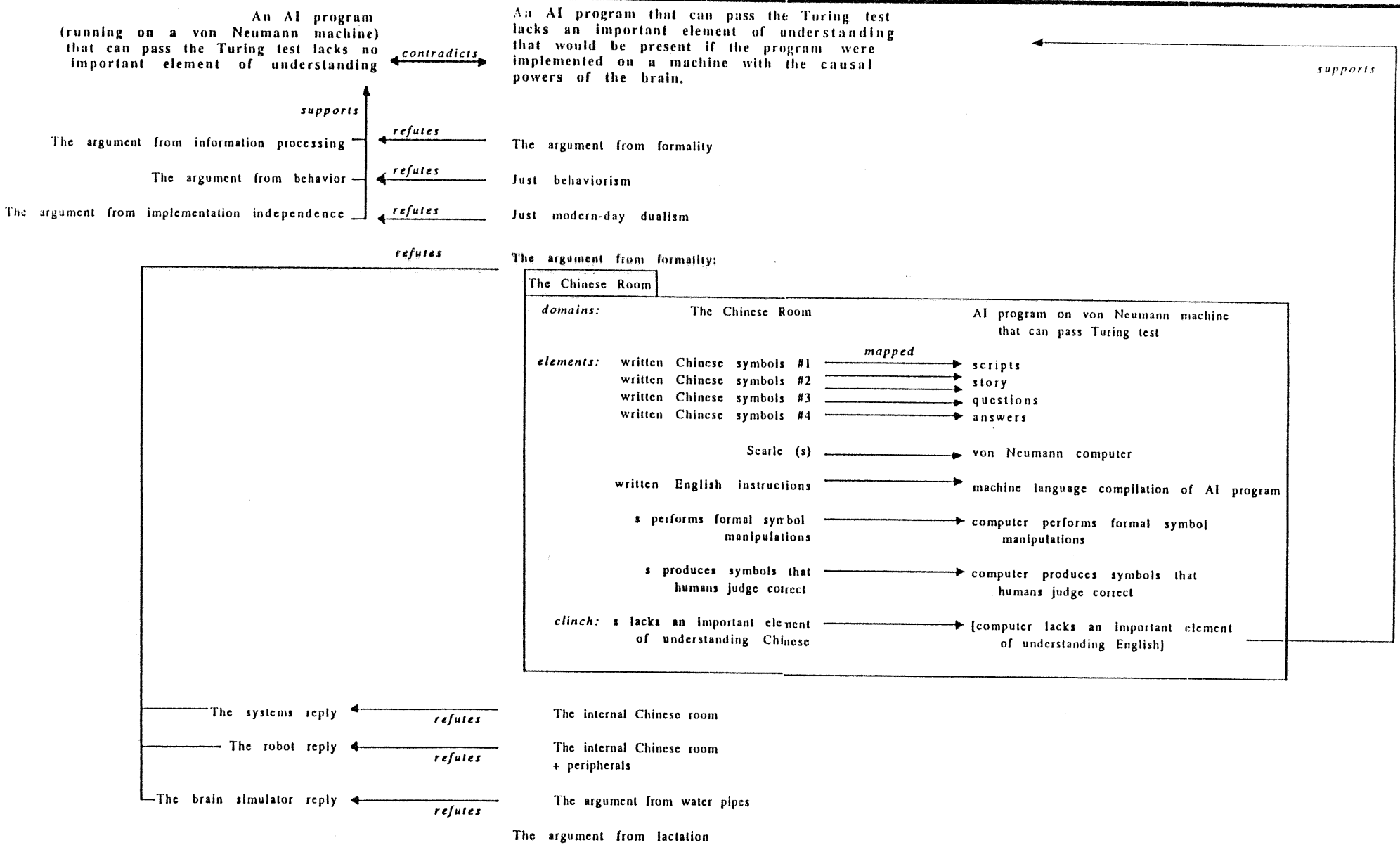

Strong AI
Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding
contradicts
An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine with the causal powers of the brain.
supports
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
mapped
elements:
written Chinese symbols #1
written Chinese symbols #2
written Chinese symbols #3
written Chinese symbols #4
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]
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.

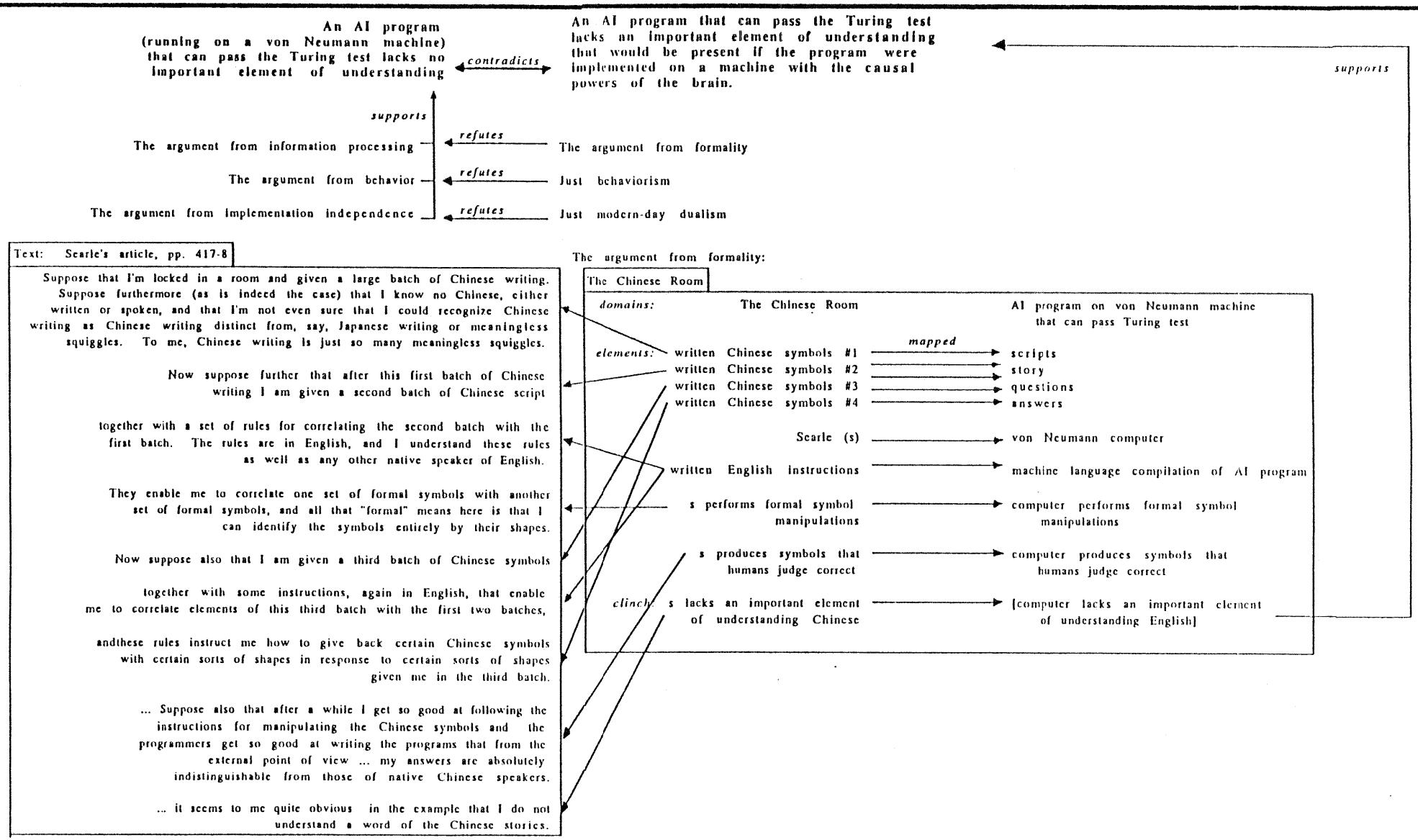

Strong AI
Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding
contradicts
An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine 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
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.
The argument from formality:
The Chinese Room
domains:
The Chinese Room
AI program on von Neumann machine that can pass Turing test
mapped
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]

Figure 4.

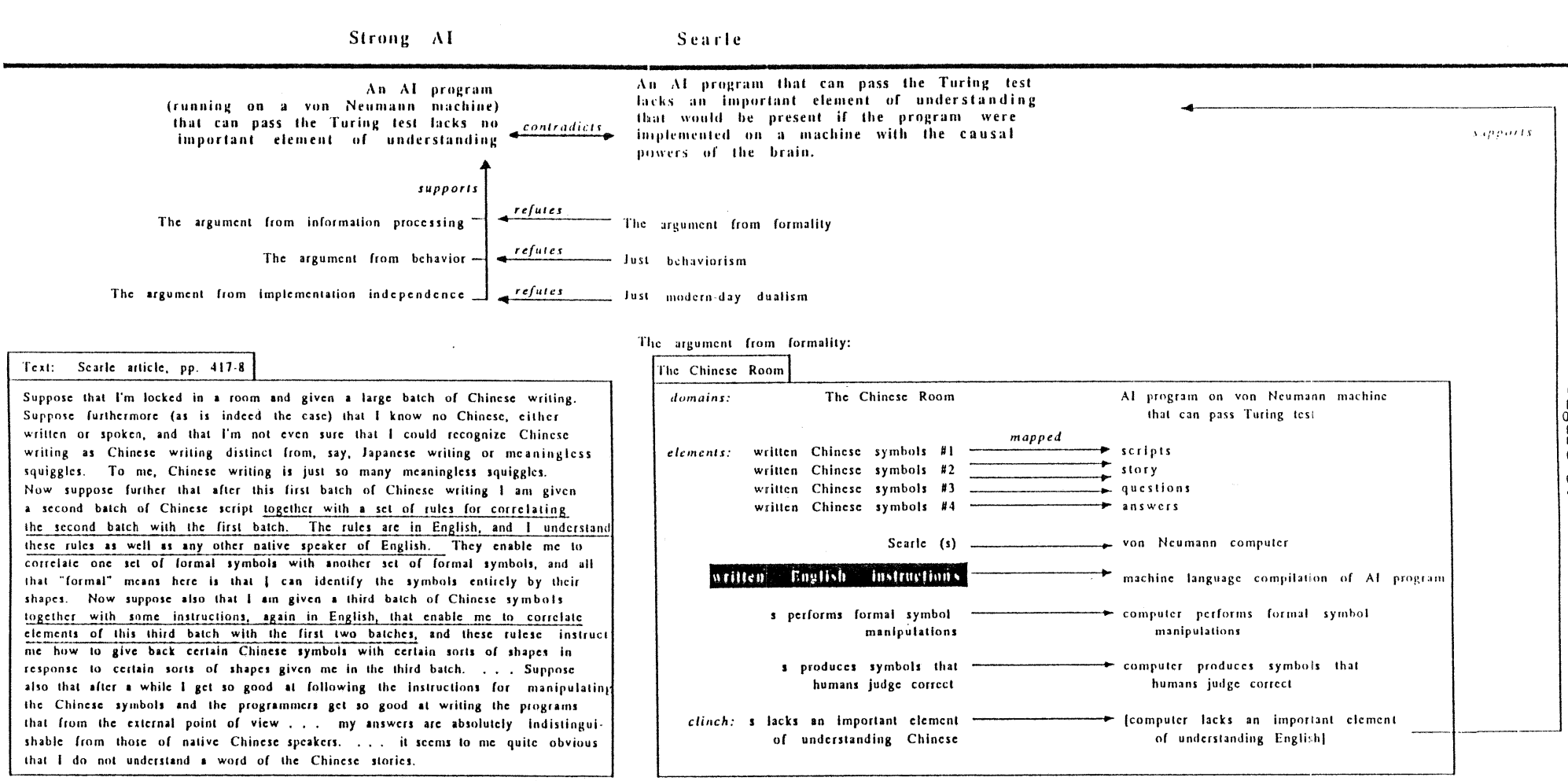

Strong AI
Searle
An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding
contradicts
An AI program that can pass the Turing test lacks an important element of understanding that would be present if the program were implemented on a machine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

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	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]

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 6.

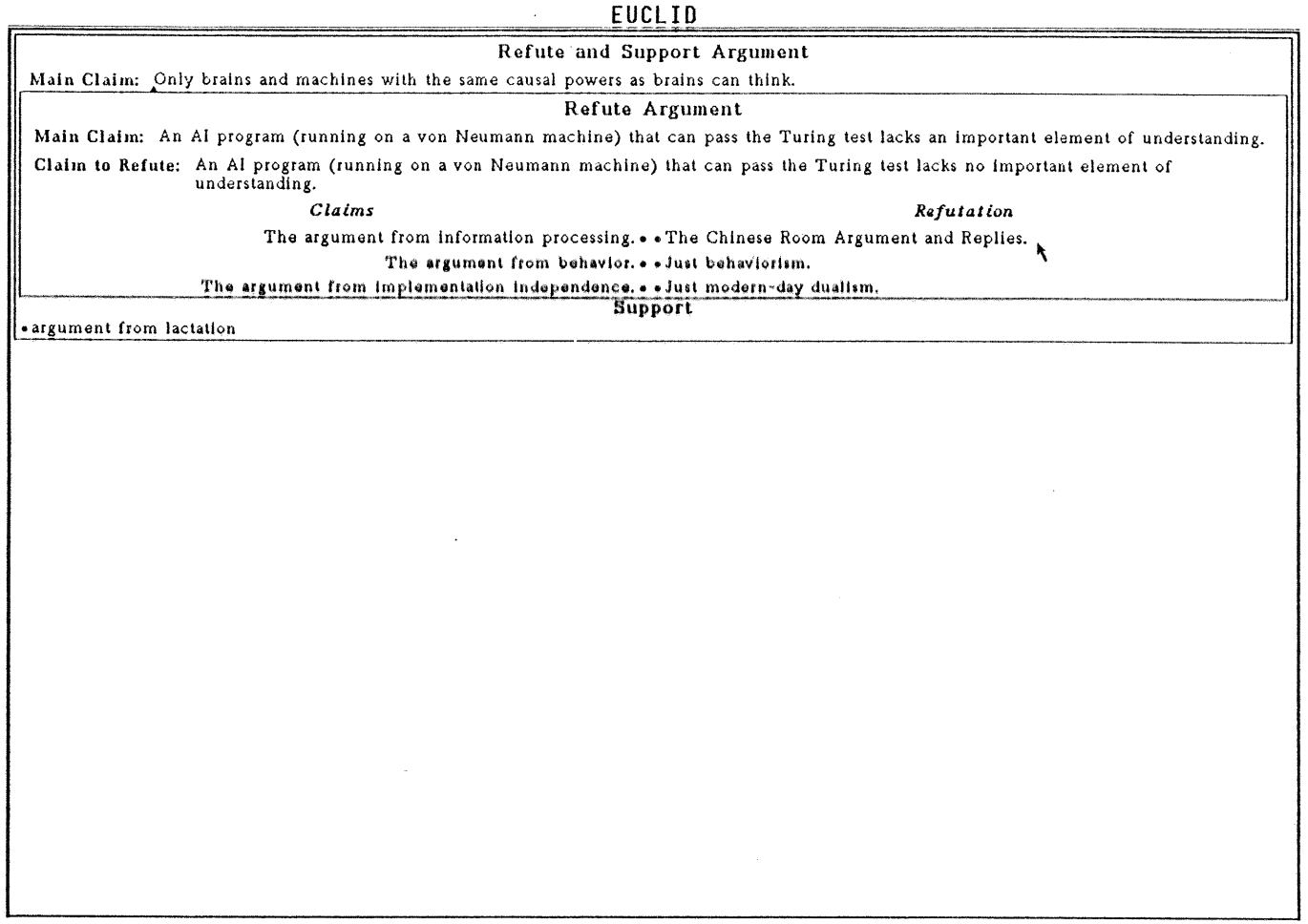

EUCLID
Refute and Support Argument
Main Claim: Only brains and machines with the same causal powers as brains can think.
Refute Argument
Main Claim: An AI program (running on a von Neumann machine) that can pass the Turing test lacks an important element of understanding.
Claim to Refute: An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding.
Claims
Refutation
The argument from information processing.
The Chinese Room Argument and Replies.
The argument from behavior.
Just behaviorism.
The argument from implementation independence.
Just modern-day dualism.
Support
argument from lactation

Figure 7.

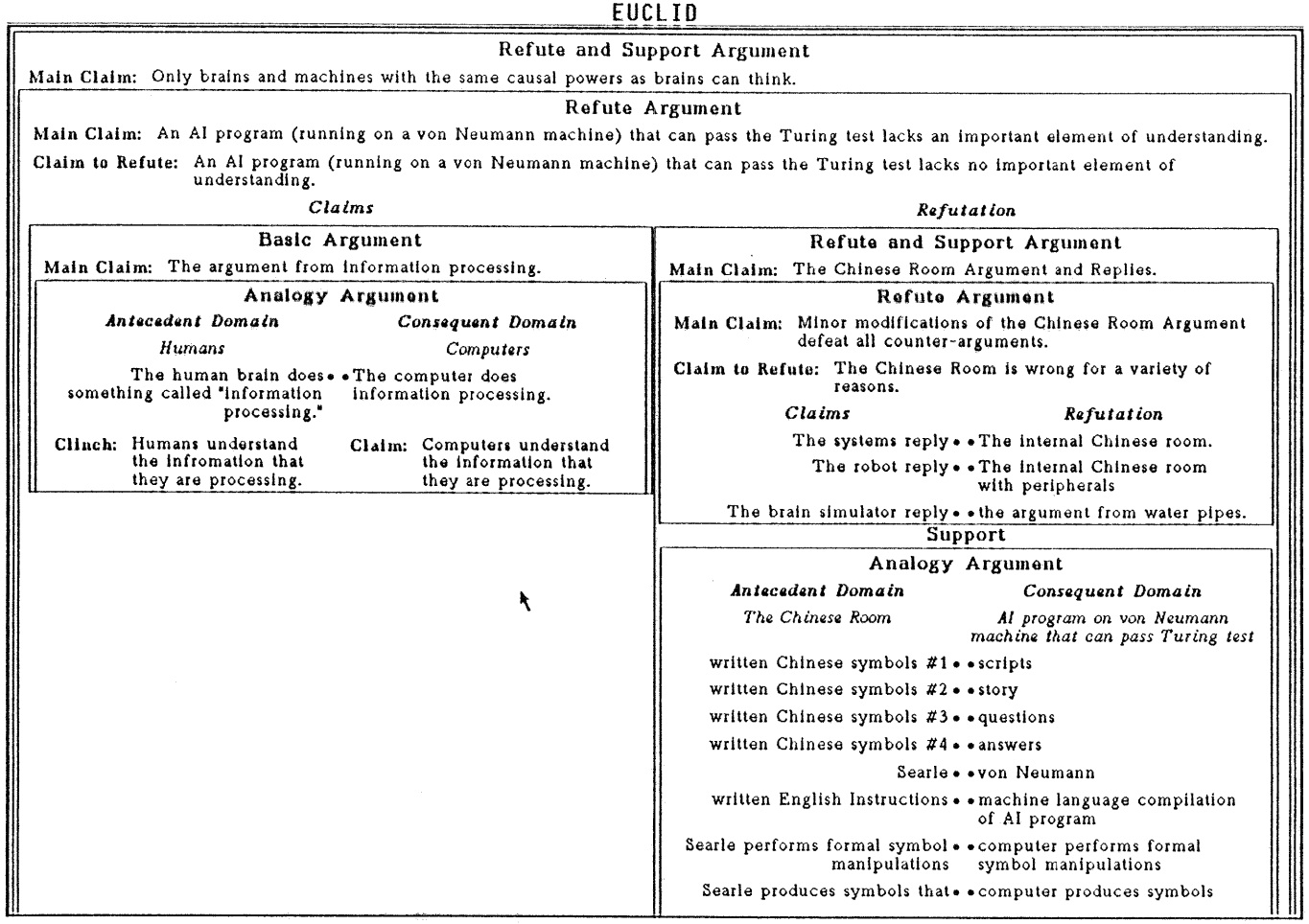

EUCLID
Refute and Support Argument
Main Claim: Only brains and machines with the same causal powers as brains can think.
Refute Argument
Main Claim: An AI program (running on a von Neumann machine) that can pass the Turing test lacks an important element of understanding.
Claim to Refute: An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding.
Claims
Refutation
Basic Argument
Main Claim: The argument from information processing.
Analogy Argument
Antecedent Domain
Consequent Domain
Humans
Computers
The human brain does something called "information processing."
The computer does information processing.
Clinch: Humans understand the infromation that they are processing.
Claim: Computers understand the information that they are processing.
Refute and Support Argument
Main Claim: The Chinese Room Argument and Replies.
Refute Argument
Main Claim: Minor modifications of the Chinese Room Argument defeat all counter-arguments.
Claim to Refute: The Chinese Room is wrong for a variety of reasons.
Claims
Refutation
The systems reply
The internal Chinese room.
The robot reply
The internal Chinese room with peripherals
The brain simulator reply
the argument from water pipes.
Support
Analogy Argument
Antecedent Domain
Consequent Domain
The Chinese Room
AI program on von Neumann machine that can pass Turing test
written Chinese symbols #1
scripts
written Chinese symbols #2
story
written Chinese symbols #3
questions
written Chinese symbols #4
answers
Searle
von Neumann
written English Instructions
machine language compilation of AI program
Searle performs formal symbol manipulations
computer performs formal symbol manipulations
Searle produces symbols that
computer produces symbols

Figure 8.

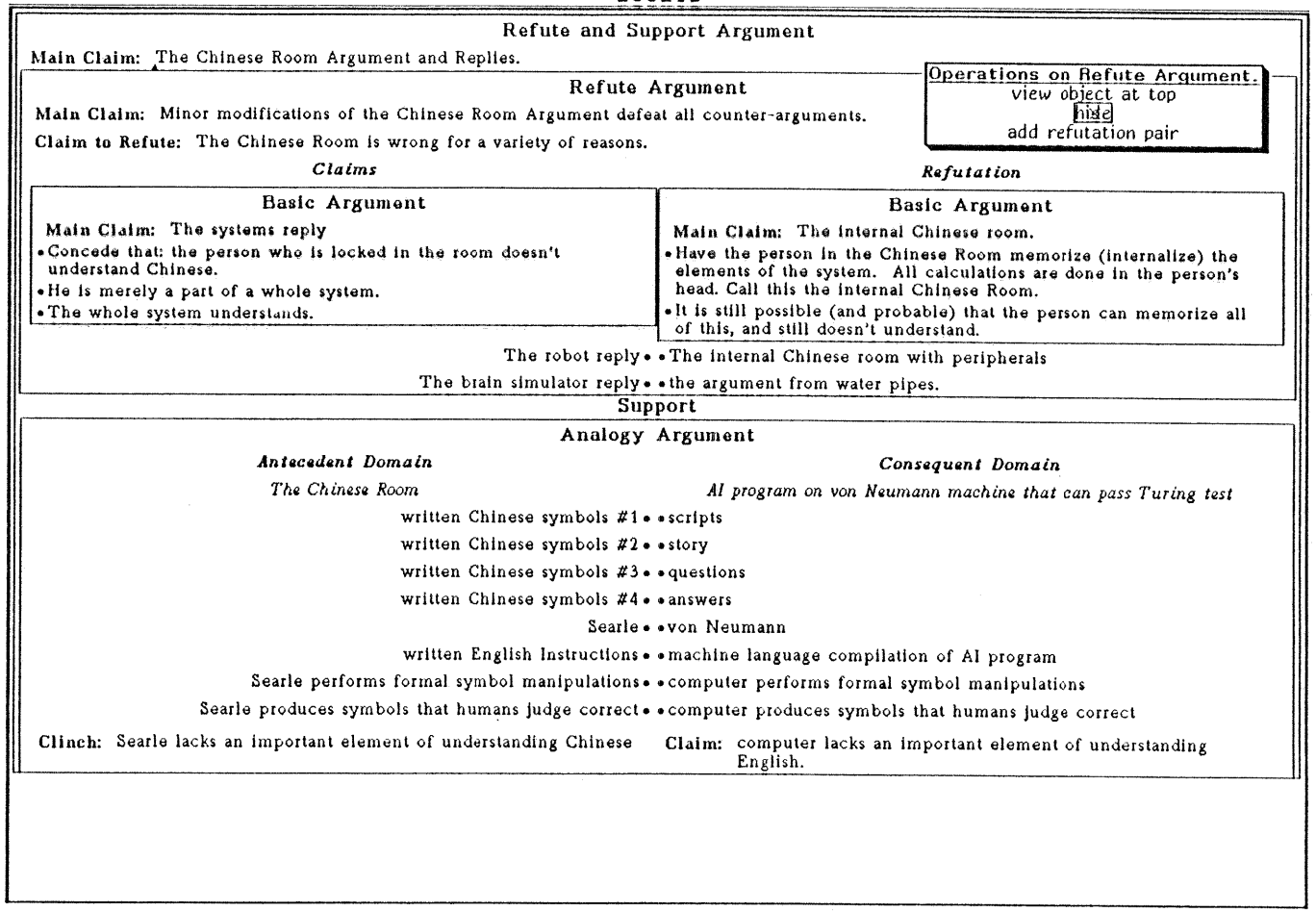

EUCLID
Refute and Support Argument
Main Claim: The Chinese Room Argument and Replies.
Refute Argument
Main Claim: Minor modifications of the Chinese Room Argument defeat all counter-arguments.
Claim to Refute: The Chinese Room is wrong for a variety of reasons.
Operations on Refute Argument.
view object at top
hide
add refutation pair
Claims
Refutation
Basic Argument
Main Claim: The systems reply
• Concede that: the person who is locked in the room doesn't understand Chinese.
• He is merely a part of a whole system.
• The whole system understands.
Basic Argument
Main Claim: The internal Chinese room.
• Have the person in the Chinese Room memorize (internalize) the elements of the system. All calculations are done in the person's head. Call this the internal Chinese Room.
• It is still possible (and probable) that the person can memorize all of this, and still doesn't understand.
The robot reply
The internal Chinese room with peripherals
The brain simulator reply
the argument from water pipes.
Support
Analogy Argument
Antecedent Domain
The Chinese Room
Consequent Domain
AI program on von Neumann machine that can pass Turing test
written Chinese symbols #1
scripts
written Chinese symbols #2
story
written Chinese symbols #3
questions
written Chinese symbols #4
answers
Searle
von Neumann
written English instructions
machine language compilation of AI program
Searle performs formal symbol manipulations
computer performs formal symbol manipulations
Searle produces symbols that humans judge correct
computer produces symbols that humans judge correct
Clinch: Searle lacks an important element of understanding Chinese
Claim: computer lacks an important element of understanding English.

Figure 9.

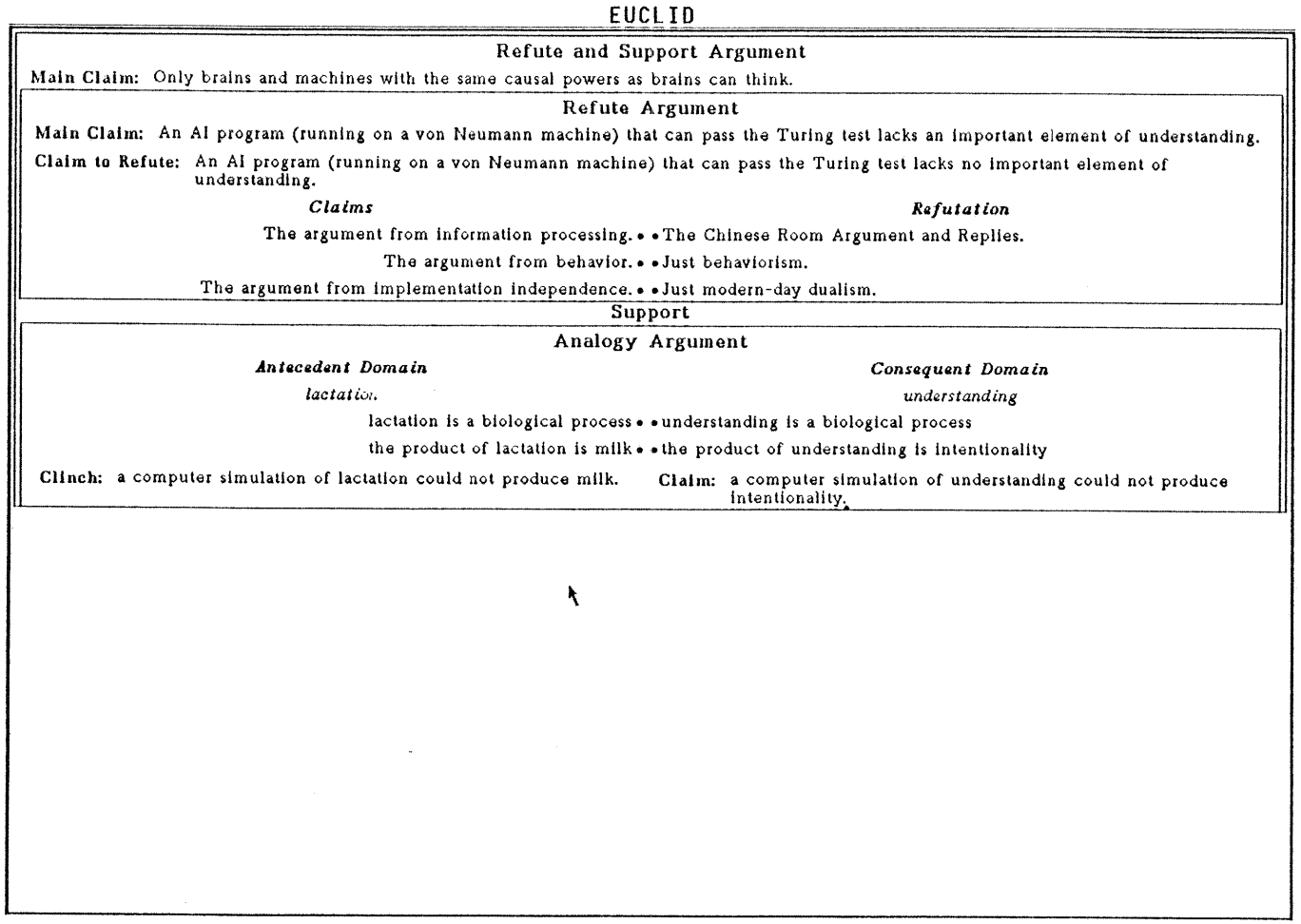

EUCLID
Refute and Support Argument
Main Claim: Only brains and machines with the same causal powers as brains can think.
Refute Argument
Main Claim: An AI program (running on a von Neumann machine) that can pass the Turing test lacks an important element of understanding.
Claim to Refute: An AI program (running on a von Neumann machine) that can pass the Turing test lacks no important element of understanding.
Claims
Refutation
The argument from information processing.
The Chinese Room Argument and Replies.
The argument from behavior.
Just behaviorism.
The argument from implementation independence.
Just modern-day dualism.
Support
Analogy Argument
Antecedent Domain
lactation.
Consequent Domain
understanding
lactation is a biological process
understanding is a biological process
the product of lactation is milk
the product of understanding is intentionality
Clinch: a computer simulation of lactation could not produce milk.
Claim: a computer simulation of understanding could not produce intentionality.

Figure 10.

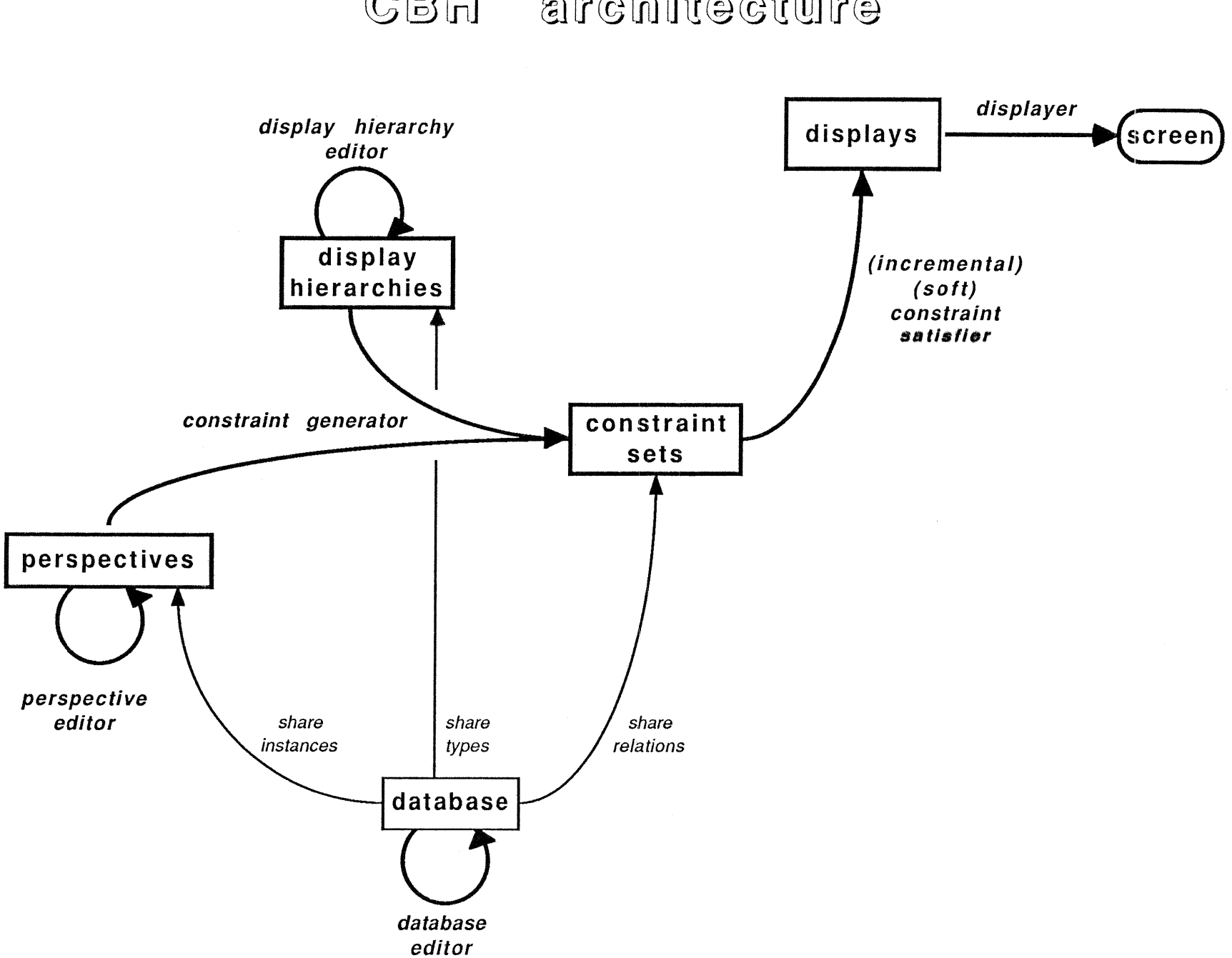

CBH architecture
display hierarchy editor
displays
displayer
screen
display hierarchies
(incremental) (soft) constraint satisfier
constraint generator
constraint sets
perspectives
perspective editor
share instances
share types
share relations
database
database editor

Figure 11.

Figure 12.

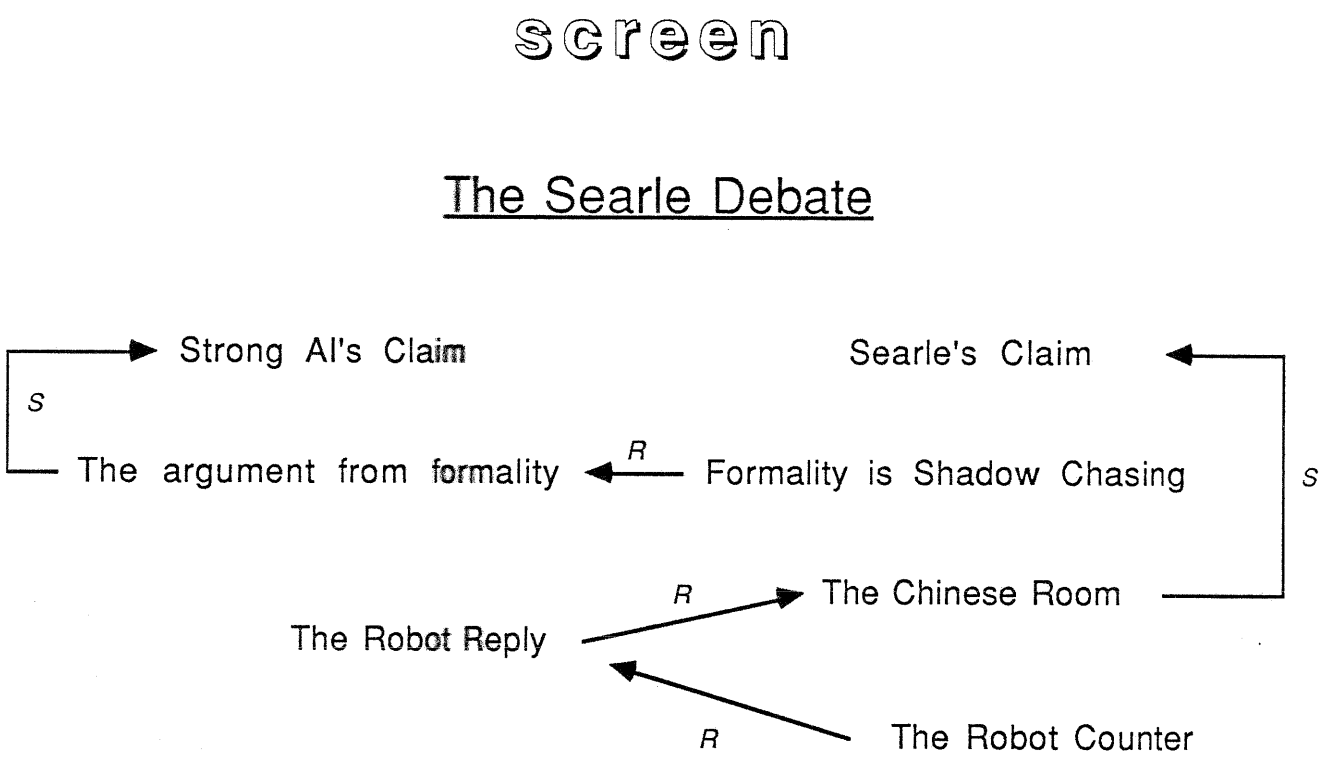
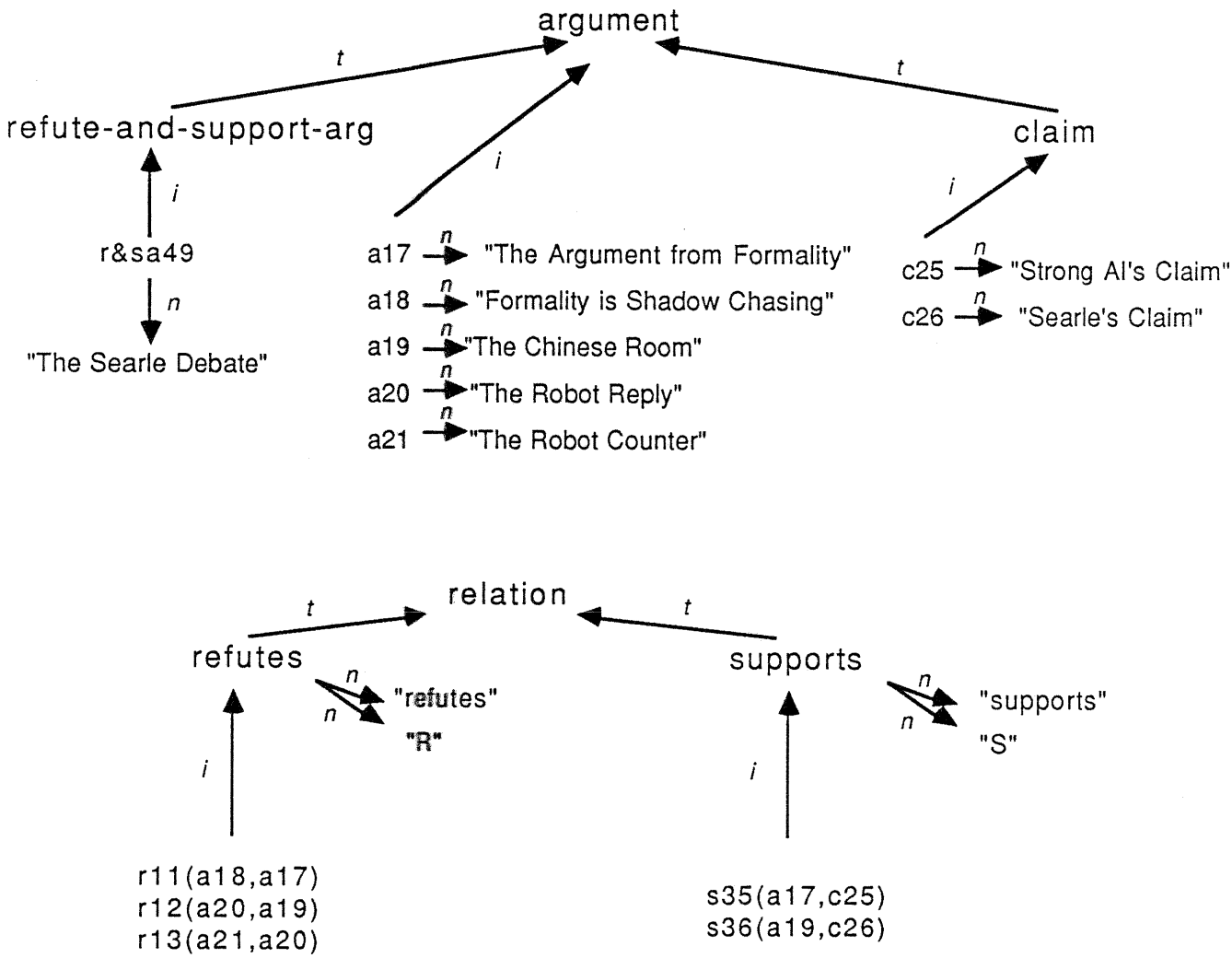

screen
The Searle Debate
Strong AI's Claim
Searle's Claim
S
The argument from formality
R
Formality is Shadow Chasing
S
R
The Chinese Room
The Robot Reply
R
The Robot Counter

Figure 13.

database



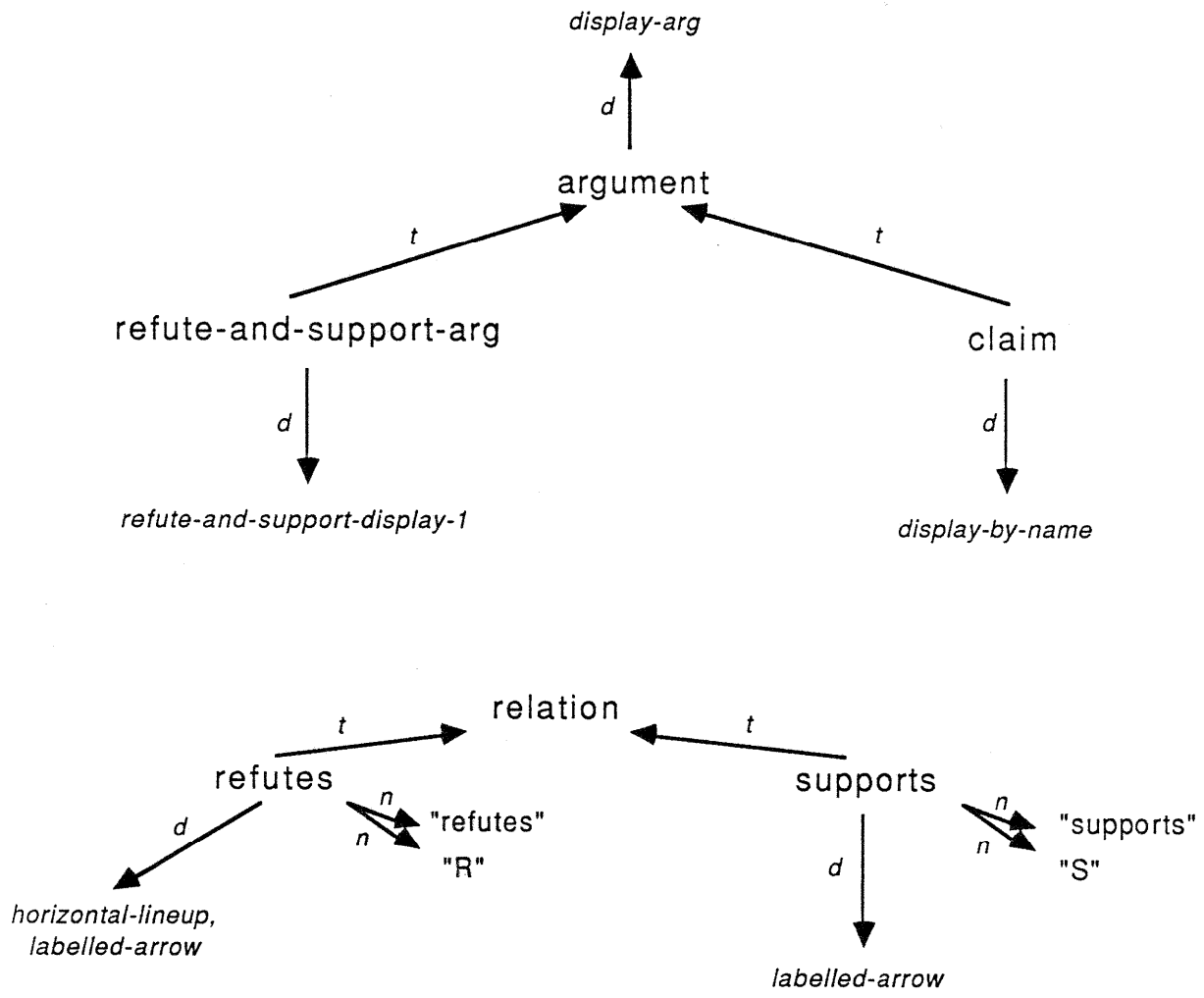
relations: **t = type-of** **n = has-name** **i=instance-of**

active perspective

r&sa49, a17 - a21, c25 - c26, s35 - s36, r11 - r13

Figure 14.

display hierarchy



relations:

d = display-constraint-schema
t = type-of
n = has-name
i=instance-of

Figure 13:

Display Constraint Schemata

```
display-by-name (X,Rect,Printstring) :=
        contains-entire-string(Rect, Printstring),
        name(X, content(Printstring)),
        font(Printstring) = default-font(type(X)),
        size(Printstring) approx=(3) default-size(type(X)) .

refute-and-support-arg-display-1 (A) :=
        vertical-stack(R5,[R1,R4]),
        horizontal-stack(R4,[R2,R3]),
        display-by-name(A,R1,Printstring),
        underlined(Printstring),
        display-arg(refute-side(A),R2),
        display-arg(support-side(A),R3) .

display-arg (A,R) :=
        vertical-stack(R0,[R1,R2,R3]),
        contains-entire-string(R1,Printstring1),
        name(A, contents(Printstring1)),
        display-claim(R2,conclusion(A))
        display(R3,arg-body(A)) .

vertical-stack (R,Rlist) :=
        rectangle(R),
        list-of(rectangle,Rlist),
        top(R) = top(first(Rlist)),
        bottom(R) = bottom(last(Rlist)),
        for-all(Ri,tail(Rlist),top(Ri)=bottom(predecessor(Ri,Rlist))) .
```