Author's Argumentation Assistant (AAA): A Hypertext-Based Authoring Tool for Argumentative Texts

Wolfgang SCHULER† and John B. SMITH‡

† *Integrated Publication and Information Systems Institute*
GMD-IPSI
P.O. Box 104326
D-6100 Darmstadt
Federal Republic of Germany

e-mail: schuler@darmstadt.gmd.dbp.de

‡ *Department of Computer Science*
University of North Carolina
CB # 3175 Sitterson Hall
Chapel Hill, NC 27599-3175
USA

e-mail: jbs@cs.unc.edu

ABSTRACT : We present the conceptualization and implementation of AAA, a prototype authoring tool for creating argumentation-based hyperdocuments. AAA is part of a more comprehensive effort of GMD-IPSI, where the hypertext authoring system SEPIA (Structured Elicitation and Processing of Ideas for Authoring) is developed. AAA will be used for writing and design experiments the results of which will be used in the design of SEPIA. It is designed to support the creation of argumentation patterns in accordance with the IBIS/PHI (Procedural Hierarchical IBIS) model combined with a micro argumentation structure according to Toulmin. For rapid prototyping purposes it has been implemented as a hypertext system using the Writing Environment WE developed at UNC.

AAA uses a combination of different independent but cooperating modes of operation dedicated to different cognitive tasks of the argumentative writing process. The entire argumentation structure is represented as a layered network of typed nodes and links in which different layers correspond to different levels of abstraction.

KEY WORDS : authoring system, hypertext application, argumentation model

1 Introduction

Writing complex documents consists in many cases of providing arguments for positions or statements made in the document. The authoring activity of writing argumentative texts as well as the related document type often has

been studied. Furthermore, argumentative texts exhibit structures which offer excellent starting points for the design of an active, knowledge-based authoring and idea processing tool for creating and revising hyperdocuments. It is the design objective of the system SEPIA (Structured Elicitation and Processing of Ideas for Authoring) to represent the major portion of the functionality of an author's workbench in future hypertext environments. The main ideas underlying SEPIA are fully described in [Streitz et al. 89]: writing a document is conceived as travelling through different 'activity spaces' which usually can be associated with different system modes. In order to test the assumptions related to the argumentative writing of hyperdocuments, it was decided to develop an early prototype based on an available authoring support tool. Making use of a cooperation between GMD-IPSI and the Department of Computer Science at the University of North Carolina (UNC) we chose to use the Writing Environment (WE) for this purpose.

The implementation of AAA extends the concepts of modes and system functions expressed in WE. AAA offers different special writing modes which the author can use in order to create a final hyperdocument or a traditional linear document. In this paper the concepts of AAA and some aspects of WE are described.

2 Writing Environment WE

At UNC the TextLab research group developed a hypertext-based Writing Environment (WE) in order to support the writing process in accordance to theoretical considerations as well as to study writers' individual strategies as they use the system [Smith et al. 87]. The system provides the author with four system modes. These modes correspond to the combination of six from seven cognitive writing modes which are the result of theoretical investigations.

2.1 Cognitive Modes and Strategies

The theory underlying WE is based on the ideas of cognitive modes and the strategies individuals use to engage their various forms of thinking [Smith & Lansmann 88]. A cognitive mode is defined to be an interdependent construct of goals, intellectual products, cognitive processes, and constraints. Specific modes consist of specific combinations of these four constituents: if one changes a component, one changes modes.

This general theory is based on seven basic cognitive modes which are used by many writers of expository linear documents: exploration, situa-

tional analysis, organization, writing, editing global organization, editing coherence, editing expression.

2.2 Modes of the Writing Environment

Six of the seven cognitive modes for writing are supported by four system modes in WE. To every mode a fixed sub-window is associated. WE (Version 1.04) is implemented in Smalltalk 2.3.

The *network mode* is to be used for exploration. The underlying data model in this mode is a directed graph embedded in a two-dimensional space. Thus, the user has maximum flexibility to represent concepts as nodes (boxes with a word or phrase to express the idea), move them manually to form clusters of loosely related ideas, and link them to denote more specific relationships.

To build the actual structure of the document, the system provides a *tree mode* in which the user is constrained to create a hierarchical structure. While users could have continued working in the network mode, they are encouraged to shift to a mode specifically intended for organization, thereby encouraging them to transform their (loose) network of ideas into a well-formed hierarchical structure. At anytime in the process they can open a node and write or edit a block of text that will be associated with that node. This is done in the *editor mode*, a conventional text editor.

Finally the *text mode* is intended for coherence editing. The tree represents the structure of the linear document and the logical sequence of nodes or blocks of text that comprise it. Text mode constructs a linear form of the implied text by stepping through the tree and displays it for editing.

Thus, the four modes correspond to exploration, organization, writing, and coherence editing. For structure editing, writers use the tree mode. To support expression editing, writers may use either editor or text modes. Thus, six of the seven cognitive modes for writing are supported by the four system modes in WE. At present, WE does not support the situational analysis mode.

3 Author's Argumentation Assistant AAA

Since the SEPIA system is intended to support argumentation in hyperdocuments the experimental system AAA requires more special system modes for argumentation and rhetorical reorganization which WE does not offer. AAA can be regarded as a dedicated application and further development of WE. The main objective of AAA is to apply a specific argumentation

method to a writing model using different modes of operation. Another objective is to perform argumentative writing experiments with AAA using WE's powerful protocol and replay components.

3.1 Conceptual framework for argumentation

AAA is an experimental hypertext authoring system designed to support the generation of specific 'argumentative rhetorics'. Our approach is to combine the IBIS/PHI argumentation model [McCall 89] as macro structure with a micro argumentation structure according to [Toulmin 58]. The general combination of a macro structure with a micro structure has been proposed by [Kopperschmidt 85], but it has never been implemented and tested. Other attempts to support argumentation are EUCLID [Smolensky et al. 88], gIBIS (Graphical IBIS) [Conklin & Begeman 87] and JANUS [Fischer et al. 89]. The system SEPIA will use the Toulmin micro structure [Streitz et al. 89].

3.2 PHI (Procedural Hierarchical IBIS)

In AAA, the author can build a fixed argumentation structure applying a macro structure as the main organization principle and combining it with a micro structure. The macro structure is based on PHI (Procedural Hierarchical IBIS) [McCall 89], which is an elaboration of Rittel's IBIS (Issue Based Information System) [Kunz & Rittel 70]. PHI uses as basic elements (node types) issues, positions, arguments, and facts connected by specific link types. Similar node and link types are described and realized in gIBIS and JANUS. The nodes usually contain texts; however, this is no limitation of generality. For example, the PHI-based design environment PHIDIAS [McCall et al. 90] integrates CAD graphics into dynamic hypertext. The final result of argumentation will be a network of issues with positions and corresponding arguments as well as additional facts. The network generated in gIBIS can have isolated subgraphs while that of PHI is a connected directed graph. Having the latter graph type guarantees that the author cannot get lost in PHI and that efficient browsing mechanisms can be defined.

The basic node types of PHI are 'issues', 'positions', 'arguments'. They are fundamental, defining a specific argumentation method. The main elements are issues which are problematic topics to be handled, in PHI rigorously described as questions. To solve these questions positions are generated as claims for or against which arguments are raised by means of the relations 'support' and 'objection'. The positions are connected with issues by the link 'answer'. Additionally, there are 'facts' which can be used with

the link 'reference' to support every other node type.

The argumentation process in PHI begins with a starting top issue, which is usually the name of the application or the main title. This issue is broken down in a top-down manner into subissues serving to handle the main issue. In order to limit the number of issues the author should formulate only subissues which contribute to the solution of its higher issue and consequently also to the top issue. This is the operational concept of the 'serve' relation. The issues are again subdivided into subissues with the 'serve' relation and so forth. They can also be interrelated or substituted. The positions as well as the arguments can also be refined, using the relation 'contributes', which creates subpositions or subarguments. When doing this, a subposition functions as argument to the super-position, and vice versa; a super-argument functions as position. Positions and arguments change their semantic role, and therefore also their type, when the relation 'contributes' is applied.

These link types are necessary for using the PHI-method. For the process of argumentation, however, some additional link types have often been suggested [Conklin & Begeman 87]: issues may be replaced by other more suitable ones, and from any other node type a new issue can be suggested. The contents of two nodes with the same node type, i.e. a position, can contradict each other; logically they are the negation of each other. All suggested link types are designed for a single author environment.

3.3 System Modes and their operations

The basic concept of AAA is that of modes as defined in WE, which are independent entities (subsystems). A mode uses a dedicated window being a fixed area of the screen. AAA supports five activities of the argumentative writing process in different operational modes. The modes are called: argumentation, rhetorical, tree, text, and edit mode. In comparison to WE, the argumentation mode and the rhetorical mode substitute WE's network mode. Both new modes are specialized network modes, enriched by node and link types. Every mode maintains its own set of link and node types, its schemas in the form of subgraph types, as well as its own graph structure of node and link instances. In the same manner other necessary modes could be added, i.e. SEPIA's content mode described in [Streitz et al. 89]. A common mode for all other modes is the edit mode of WE, which serves to edit the text of the nodes in these modes. A screendump of AAA's interface with four modes is shown in Figure 1, where the edit mode is hidden. The "Chinese Room" argumentation of John Searle [Searle 80] has been used here as an illustration example which resembles the example used by EUCLID.

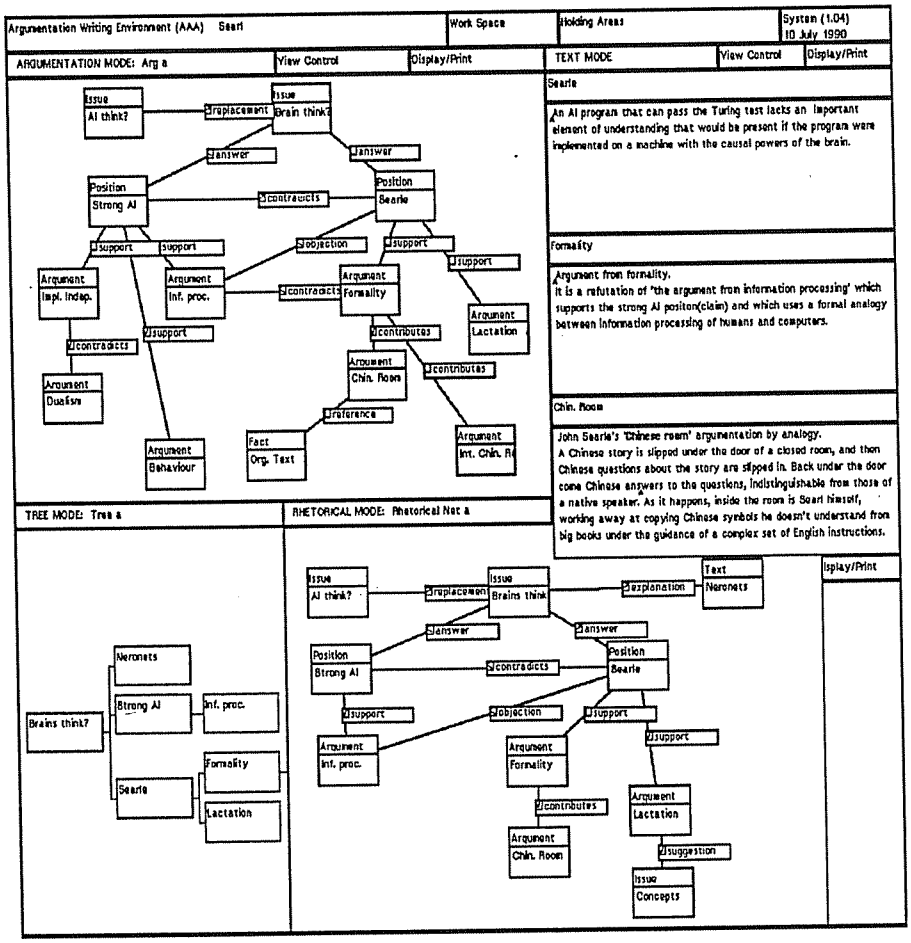

Figure 1: The Screen of AAA with its Modes

3.3.1 The argumentation mode

The argumentation mode is the nucleus of AAA, in which ideas and arguments are generated and the argumentation process takes place. The basis of this mode is the enhanced PHI structure, defining a specific method to be used by the author. The final result is an interconnected network of issues linked with sub-relationships to positions, arguments, and facts as their semantically dependent node types. It is very important for its later use that the network has no isolated graphs. Beginning from the top issue, the structure can be handled like a hierarchical one. Thus, graph levels can be defined, primarily with respect to issues, however also with respect to positions and arguments. The levels are defined by the number of nodes following the particular path beginning at the top issue and defined by the

directed links. This fact allows step by step travelling through the structure, which can be used for browsing and for traversing up and down the levels (like generalizing or specializing).

Using the basic *node types* 'issues', 'positions', 'arguments', and 'facts', the *link types* as directed links are defined as predicates (Figure 3):

- serve (A, B) - issue B serves issue A
- replacement (A, B) - issue B is a replacement of issue A
- suggestion (X, A) - issue A is a suggestion/question by X with X = {issue, position, argument, fact}
- answer (A, B) - position B is an answer to issue A
- objection (A, B) - argument B is an objection to position A
- support (A, B) - argument B is a support of position A
- contributes (A, B) - B contributes to A where {A,B} are positions or arguments (B is sub of A)
- reference (X, Y) - fact Y is a reference of X with X = {issue, position, argument, fact}
- contradicts (A, B) - A contradicts B (symmetric relation) with A, B = {issue, position, argument, fact}

The nodes and links are generated and their types selected by usual creation operations 'create node/link'. Fundamental operations on a selected node or link are change type, delete. Together with the node also its links are deleted, in a manner that disconnected graphs are avoided by appropriate operations. Also the deletion of links is checked to maintain graph connectedness if possible.

With these basic operations the argumentation net can be constructed step by step. However, there exists a set of more powerful operations to assist the process of argumentation. They operate on a selected node and create a combination of an appropriate link with a target node, or they even create a whole subgraph. In general, they offer a link-node couple depending on the type of the selected node.

Examples of operations are:

- generalize A create for A a super B where {A,B} are issues with the 'serve' link from B to A or positions, arguments with a 'contributes' link from B to A
- specialize A create for A a sub B (it is the inverse of 'generalize A')

• support position A	create an argument and connect it to position A with a 'support' link
• objects_to position A	create an argument and connect it to position A with an 'objection' link
• negate A	create B of type A and link it to A with the 'contradicts' link
• add-subgraph to A	create a subgraph of predefined type and link it to node A with a selected link type
• justify a position	create a Toulmin justification (see below)

The operation 'objects_to a position A' can also be realized without using the 'objection' link. This can be accomplished by the operation 'support (negate A)', which means that a new position B is created using the operation 'negate A', to which then the 'support' operation is applied. This is important in order to integrate the Toulmin structure in the PHI structure.

3.3.2 The Toulmin Schema

Since we are dealing with common sense argumentation instead of formal logical reasoning, an argument is more readily accepted if one can provide reasons which legitimate the 'support' or 'objection' relationship. In order to achieve this, a position with an argument can be further elaborated by using an argumentation schema proposed by Toulmin [Toulmin 58] which is used and described in [Streitz et al. 89]. Its node types are: datum, claim, warrant, backing, and rebuttal. The main link type 'so' is defined as 'so (datum, claim)' and links a 'datum' with a 'claim'. The 'so' conclusion of an argument is accepted with a higher probability if one can provide a warrant (usually a rule) which in turn can be backed by a 'backing' using the link 'on_account_of (backing, warrant)'. Thus, there exists a link 'since' which points from the node 'warrant' to the link 'so', representing a predicate of second order: 'since (warrant, (so (datum, claim))'. This means that a link owns a link pointing to it, which cannot be represented directly as a WE data type. In AAA this is internally represented by a special link type pointing from 'warrant' to both 'claim' and 'datum'. The 'claim' can be further questioned by a 'rebuttal' based on the link 'unless (rebuttal, claim)'. The Toulmin schema is shown in Figure 2.

The combination of the Toulmin schema with PHI can be achieved if the 'claim' is identified with 'position', and 'datum' with 'argument'. Both are considered as part of this more detailed micro structure. The 'so' link can be both the 'support' or 'objection' link of PHI. However, if the 'objection' link is replaced by the combination of a 'support' link with the 'contradicts'

link as descibed above, the 'so' can uniquely be identified with 'support'. The 'so' link can then be further elaborated to be a part of the whole Toulmin structure by using the operation 'justify' which activates the Toulmin schema. The resulting total argumentation structure is defined as the intersection set of the node and link types defined by PHI and the Toulmin schema.

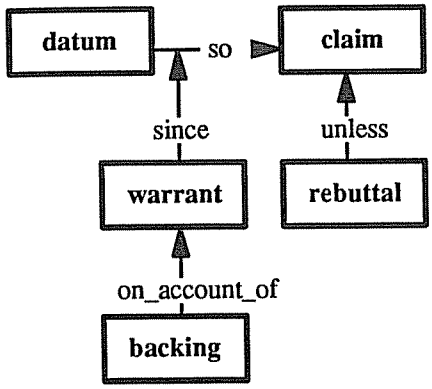

Figure 2: The Toulmin Argumentation Schema

Operations for copy/paste from the argumentation to the rhetorical mode rely on predefined subgraphs:

- copy the whole argumentation network
- copy a tree substructure
- copy an issue group: issue with all its positions with their arguments and with facts
- copy the issue structure up to level n (the possibility above is level 1)
- copy a positionargument with a Toulmin schema
- copy a set of manually selected nodes
- copy a single node
- copy a user defined subgraph
- copy a user defined combination of subgraphs, a higher subgraph.

3.3.3 The rhetorical mode

Structure and content of a non-linear (or linear) argumentative document are organized in the rhetorical mode. This mode enables the author to select appropriate elements or substructures of the argumentation mode and to reorganize them according to his or her rhetorical objectives and principles. The elements can be modified and new text elements can be added. For these

rhetorical purposes the mode offers specific operations as well as additional node and link types. Such new node and link types may serve for example to attach explanations or simple text additions to the arguments.

Figure 1 displays in the lower right side the result of a first step of such a reorganization process: a manually selected subgraph of the argumentation graph has been copied into the rhetorical mode and a rhetorical item has been added using the special rhetorical node type 'Text' and link type 'explanation'. The graphical display has been rearranged, too.

In a further step an author will transform at least some (or may be all) of the argumentative types into appropriate rhetorical ones. The used argumentative schemas will be transformed into appropriate rhetorical argumentation schemas which thereafter will be completed by the author adding new contents (i.e text or graphics) and revising existing contents. The rhetorical nodes themselves will usually aggregate argumentative subgraphs or schemas into one coherent text chunk which then may have pointers to the original argumentative basis according to the author's intention. Then, the author will have to design the final hypertext document by defining i.e. reading paths and display attributes. All these operations as well as the organization of the rhetorical mode mainly depend on the model of the intended hyperdocument as a product of the system. A proposal of a general hyperdocument structure is currently being developed.

Which link types are appropriate for rhetorical purposes mainly depend on the model of rhetorics used. We have elaborated a hierarchical system of label types with three levels of specialization. An unspecified first level type 'reference' can be detailed by using one of the second level types: explanation/generalization, consequence, comparison, illustration, specification, comprehension. The third level contains even more refined link types. This type system is being experimentally tested.

3.3.4 Submodes for the definition of node types, link types, and schemas

In order to define and display the types of nodes and links as well as subgraph schemas the argumentation mode (and also the rhetorical mode) provide two special dedicated technical submodes. In the *type definition mode* the basic node and link types can be defined as graphical items. The types can be created, handled and arranged there in a similar manner as in its super mode. In addition to the predefined types which constitute the application model used, the user can also define his own types. The text content of a node will contain comments about its type defintion or usage. The result is a graph representing the defined node and link types together with their

constraints and further information. Thus, the type definition mode serves three tasks: definition of the application types, addition of user types, and graphical presentation of the type structure as user help.

The *argumentation type definition mode* with its type graph is shown in Figure 3. The upper part of the graph contains the basic PHI node types 'Issue', 'Position', 'Argument', and 'Fact' together with their link types listed above. The types defining the Toulmin schema are shown in the left bottom section. In the right section some additional 'general' node types 'Author', 'Date', and 'Text' are defined. Their link types can be connected to every other node type. While the node types 'Author', 'Date' can be useful for every mode the node type 'Text' with its link types 'explanation' etc. are only used in the rhetorical mode. Such types will have to be defined in the rhetorical type definition mode. For test purposes, the graph of Figure 3 defines the types for both modes.

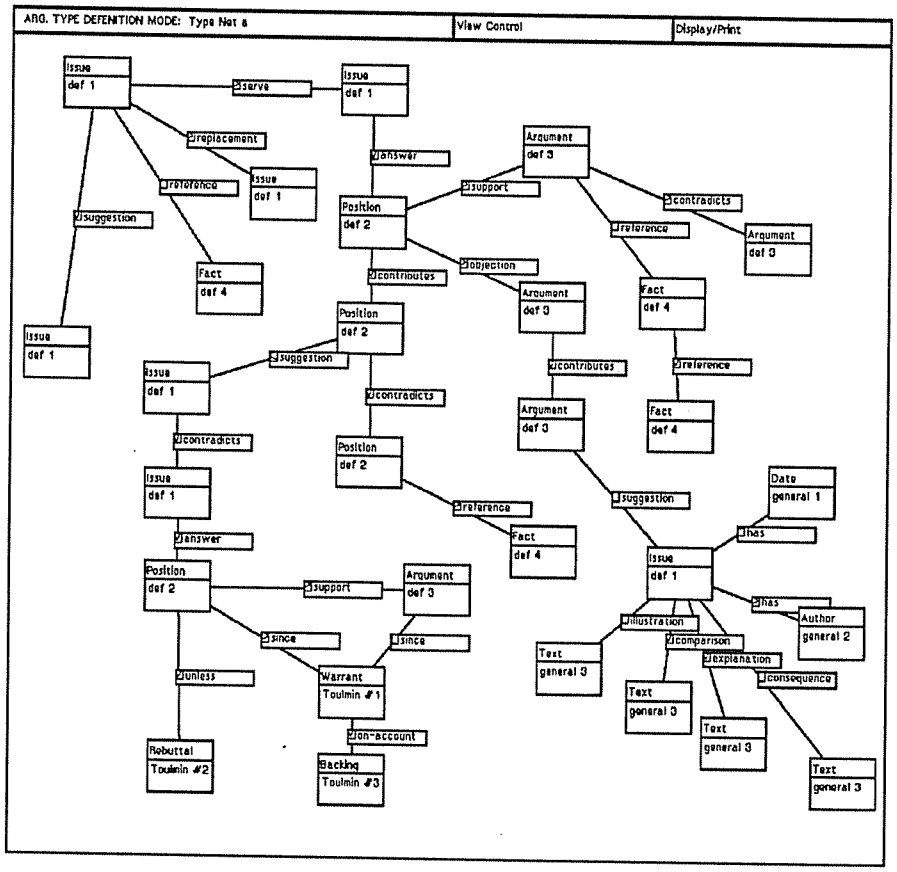

Figure 3: Type Definition Submode of the Argumentation Mode

Along with the type definition mode a *schema definition mode* is planned which will offer the possibility to define arbitrary subgraph types constituting network substructures being constructed of the allowed node and link type set. A subgraph type is defined by constructing a normal connected graph like in another mode. It is labelled like a node or link type and can be considered as a complex node. Such a complex node can itself be used in order to construct other higher subgraphs. So a subgraph type will be used together with normal node and link types or with other subgraph types in order to construct higher subgraph types. Also a recursive subgraph type definition should be possible.

Subgraph types are used as schemas to assist easy input using prompts, browsing and display, and for the definition of larger output structures, as well as for copying from one mode and pasting to another. Like the atomic types, the schemas as subgraph types can also be predefined or they are defined by the user. The latter case is very important for the formulation of a query in form of a subgraph type. In most other operations a subgraph type is applied to a selected target node.

3.3.5 Other modes

The other modes are the *tree and text mode.* They are the same as in WE and serve to define a linear document if this is the author's intention. The tree mode is based on the single-link type super/sub, which means that all information about link types in the original structure of the rhetorical mode is lost. For this purpose the rhetorical mode can additionally serve to identify and organize substructures which ultimately shall be elements of an intended hierarchical outline. The resulting rhetorical document can be reorganized into a tree structure in the tree mode with the single link type sub/super. Restructuring is supported by two basic WE operations to move substructures from the network mode into the tree mode. The result in the tree mode can then be transformed into a linear document in the text mode where the whole text is displayed and individual text elements can be edited. However, instead of using the rhetorical mode to create a linear document it can also be the basis of a hyperdocument.

Figure 1 shows a rhetorical graph in the rhetorical mode which has been transformed into a tree structure in tree mode. In the tree mode a subtree with the node 'Searle' has been selected. It contains a part of the 'Chinese Room Argument' and is linearized into scrollable text in the text mode.

3.3.6 Operations between the modes

Between the different modes, appropriate transformation operations have to be available including cut/copy/paste. The main flow of node/link data substructures will be along the modes: argumentation → rhetorical → tree → text. Very important in AAA is the transfer of subgraphs between these modes, especially from the argumentation mode to the rhetorical mode. These subgraphs can be, for example, single nodes, individually selected nodes, subtrees and schemas like the Toulmin schema.

The main mechanism to transport subgraphs manually from one network mode into another uses the copy/paste operation. First, the user selects in the source mode the subgraph type. The subgraph types as structural schemas have either been already predefined in the system, i.e. the Toulmin-schema, or they can be defined individually by the user. Every subgraph type is identified by a unique name. Then, having selected a specific node type as the root, a special subgraph type can be selected from the set of subgraph types which are applicable to this node type. If the node is part of an existing subgraph of the selected type (usually the root), this subgraph is copied to a cyclic buffer containing all copied subgraphs. The subgraphs copied from the source mode into this buffer can then be pasted to other selected nodes in the destination mode.

4 Tools for experiments in writing

In order to test the theory of cognitive modes, to study writers' cognitive strategies, and to refine the WE system on the basis of empirical results, WE has been designed to be used as an instrument with which to conduct empirical studies. To support those studies, TextLab at UNC developed four kinds of new *protocol analysis tools* which are used in a series of studies.

These tools comprise a *protocol tracking function* which records each user action as an event using a simple protocol transcription language. A *replay function* can take the protocol and recreate and replay the user's session. Third, a *cognitive grammar* in form of an expert system parses the protocols for a session and produces a parse tree that represents a user's strategy for this session [Smith et al. 89]. Finally, *display tools* help to interpret the protocols. These tools operate on the parse tree produced by the grammar, data derived from the parse, or statistical analyses.

AAA will offer similar protocol analysis tools as WE to create a protocol of an author's session, which then can be used for replay and analysis in order to study empiricallly the process of argumentative and rhetorical writing.

5 Conclusions

AAA is an experimental system being still in progress and designed for investigating our ideas about argumentative writing in different modes. Like WE it is built for small-scale applications with small amounts of nodes and links in each mode. For large amounts of nodes and links AAA has to be based on a reliable and efficient DBMS, i.e. IPSI's hypermedia engine HyperBase [Schütt & Streitz 90]. Further, the graphical display and handling capabilities have to be refined, i.e. for subgraphs. Other extensions are i.e. to add new modes, retrieval possibilities or to offer active helps for argumentation.

Acknowledgements

Thanks are due especially to our collegues Jörg Haake, Jörg Hannemann, Werner Rehfeld, Helge Schütt, Norbert Streitz, and Manfred Thüring working in the WIBAS project at GMD-IPSI, as well as to Gordon Ferguson, Yen-Ping Shan, and others of the UNC TextLab group. The ideas presented in this paper have greatly profited from discussions with them.

References

[Conklin & Begeman 87] J. CONKLIN and M. L. BEGEMAN, "gIBIS: A Hypertext Tool for Design Deliberation". In *Hypertext '87 Proceedings*, Chapel Hill, NC: Univ. of North Carolina, Nov. 1987, pp. 247–251.

[Fischer et al. 89] G. FISCHER, R. MCCALL, and A. MORCH, "JANUS: Integrating Hypertext with a Knowledge Based Design Environnment". In *Hypertext '89 Proceedings*, New York: ACM, Nov. 1989, pp. 105–117.

[Kopperschmidt 85] J. KOPPERSCHMIDT, "An Analysis of Argumentation". In T. A. van Dijk (Ed.): *Handbook of Discourse Analaysis*, vol. 2, *Dimensions of Discourse*, London: Academic Press, 1985, pp. 159–168.

[Kunz & Rittel 70] W. KUNZ and H. W. RITTEL, "Issues as Elements of Information Systems". Working paper 131, 1970. Berkeley, CA: Univ. of California, Center for Planning and Development Research.

[McCall 89] R. MCCALL, "MIKROPLIS: A Hypertext System for Design". *Design Studies*, 1989, vol. 10, no. 3, pp. 228–238.

[McCall et al. 90] R. MCCALL, P. BENNETT, P. d'ORONZIO, J. OSTWALD, F. SHIPMAN, N. WALLACE, "PHIDIAS: A PHI-based Design Environment Integrating CAD Graphics into dynamic Hypertext". In this volume.

[Schütt & Streitz 90] H. SCHÜTT and N. STREITZ, "HyperBase: A Hypermedia Engine Based on a Relational Database Management System". In this volume.

[Searle 80] J. SEARLE, "Minds, Brains, Programs". The Behavioral and Brain Sciences, 1980, vol. 3, pp. 417–457.

[Smith et al. 87] J. B. SMITH, S. F. WEISS, and G. J. FERGUSON, "A Hypertext Writing Environment and its Cognitive Basis". In *Hypertext '87 Proceedings.* Chapel Hill, NC: Univ. of North Carolina, Nov. 1987, pp. 195 – 214.

[Smith & Lansman 88] J. B. SMITH and M. LANSMANN, "A Cognitive Basis for a Computer Writing Environment". Technical Report TR87-032, June 1988. Chapel Hill, NC: Univ. of North Carolina, Dep. of Comp. Science.

[Smith et al. 89] J. B. SMITH, M. C. ROOKS, and G. J. FERGUSON, "A Cognitive Grammar for Writing: Version 1.0". Technical Report TR89-011, April 1989. Chapel Hill, NC: Univ. of North Carolina, Dep. of Comp. Science.

[Smolensky et al. 88] P. SMOLENSKY, B. FOX, R. KING, and C. LEWIS, "Computer-aided Reasoned Discourse or, how to Argue with a Computer". In R. Guindon (Ed.): *Cognitive Science and its Application for Human-Computer Interaction*, Norwood, NJ: Ablex, 1988, pp. 109–162.

[Streitz et al. 89] N. STREITZ, J. HANNEMANN, and M. THÜRING, "From Ideas and Arguments to Hyperdocuments: Travelling through Activity Spaces". In *Proceedings Hypertext '89*, New York: ACM, Nov. 1989, pp. 343–364.

[Toulmin 58] S. TOULMIN, *The Uses of Argument.* Cambridge University Press, 1958.