Hermes: Supporting Argumentative Discourse in Multi-Agent Decision Making

Nikos Karacapilidis
Computer Science Department, DI–LITH
Swiss Federal Institute of Technology
IN Ecublens, 1015 Lausanne, Switzerland
karacapi@di.epfl.ch

Dimitris Papadias
Department of Computer Science
Hong Kong Univ. of Science & Technology
Clearwater Bay, Hong Kong
dimitris@cs.ust.hk

Abstract

This paper describes Hermes, a system that enhances group decision making by providing an argumentation framework to the agents involved. The system organizes the existing knowledge in a discussion graph, which consists of *issues, alternatives, positions* and *preference relations.* Argumentation is performed through a set of *discourse acts* which trigger appropriate procedures for the propagation of information in the graph. Hermes is able to handle incomplete, qualitative and inconsistent information, and provides mechanisms for weighing arguments.

Introduction

Group Decision Support Systems (GDSSs) have been defined as interactive computer-based systems which facilitate the solution of ill-structured problems by a set of decision makers, working together as a team (Kreamer and King 1988). The main objective of a GDSS is to augment the effectiveness of decision groups through the interactive sharing of information between group members and the computer. This can be achieved by (i) removing communication impediments, and (ii) providing techniques for structuring the decision analysis and systematically directing the pattern, timing, or content of the discussion. Among the major issues arising during the development of such a system are the effective work organization in order to improve coordination, and the use of communication technology to make decision making more efficient. Provision of rules and procedures for achieving consistency and automation of data processing, especially in data intensive decision making situations, are also of high importance.

This paper presents Hermes, an advanced GDSS able to address the above issues. The system augments classical decision making approaches by supporting argumentative discourse among decision makers. It is implemented in Java applets and runs on the Web, thus provides relatively inexpensive access to a broad public. Hermes can be used for distributed, asynchronous or synchronous collaboration, allowing agents to surpass the requirements of being in the same place and working at the same time.

Numerous web-based conferencing systems have already been deployed, such as *AltaVista Forum Center, Open Meeting, NetForum*, and *UK Web's Focus*, to mention some. They usually provide means for discussion structuring and user administration tools, while the more sophisticated ones allow for sharing of documents, on-line calendars, embedded e-mail and chat tools, etc. Discussion is structured via a variety of links, such as simple *responses* or different *comment types* (e.g., *qualify, agree, example* in *Open Meeting*) to a previous message. However, the above systems merely provide threaded discussion forums, where messages are linked "passively", which usually leads to an unsorted collection of vaguely associated comments. As pointed out by the developers of *Open Meeting*, there is a lack of consensus seeking abilities and decision making methods (Hurwitz and Mallery 1995). On the contrary, Hermes focuses on aiding decision makers reach a decision, not only by efficiently structuring the discussion, but also by providing reasoning mechanisms for it. It is an *active* system, that constantly checks for inconsistencies and updates the discourse status.

Increasing interest also develops in implementing argumentation support systems for different types of groups and application areas. Such systems intend to address the needs of a user to interpret and reason about knowledge during a discourse. For instance, *gIBIS* (Conklin and Begeman 1987) is a workstation-based hypertext groupware tool that aims at capturing the rationale of design processes; *Euclid* (Smolensky et al. 1987) provides a graphical representation language for generic argumentation; *JANUS* (Fischer, McCall, and Morch 1989) is based on acts of critiquing existing knowledge in order to foster the understanding of design knowledge; *SEPIA* (Streitz, Hannemann, and Thüring 1989) is a knowledge-based authoring and idea processing tool for creating and revising hyper-documents that views authoring as a design process. Finally, *Belvedere* (Suthers et al. 1995) uses a rich graphical language to represent different logical and rhetorical relations within a debate. It was originally designed to support students engaged in critical discussion of science issues. Although this second category of systems provides a cognitive argumentation environment that stimulates discussion among participants, it also lacks decision making capabilities.

Argumentation and Decision Making

Most approaches to argumentative discourse follow two main methodologies, namely, formal and informal logic. According to the *formal perspective*, arguments are de-contextualized sets of sentences or symbols viewed in terms of their syntactic or semantic relationships. On the other hand, *informal logic* views arguments as pragmatic, i.e., their meaning is a function of their purposive context. Most AI researchers have focused on formal models of argumentation based on various logics. For instance, Brewka (Brewka 1994) reconstructed Rescher's theory of formal disputation (Rescher 1977), while Gordon's work (Gordon 1995), was based on Geffner and Pearl's concepts of conditional entailment (Geffner and Pearl 1992).

Argumentation elements

The argumentation framework of HERMES is a variant of the informal IBIS model of argumentation (Rittel and Webber 1973). HERMES supports as argumentation elements *issues, alternatives, positions*, and *constraints* representing *preference relations*. Throughout this paper, we use a real example about the planning of cyclepaths in the city of Bonn. The agents involved in the discussion, namely the representatives from the Cyclists Union and the related City Hall department, bring up the necessary argumentation in order to express their interests and perspectives. Figure 1 illustrates an instance of the corresponding HERMES *discussion forum*. As shown, our approach maps a multi-agent decision making process to a discussion graph with a hierarchical structure.

Issues correspond to decisions to be made, or goals to be achieved (e.g., issue-188: "selection of a cyclepath for the city"). They are brought up by agents and are open to dispute. Issues consist of a set of *alternatives* that correspond to potential choices (e.g., alternative-190: "select the already planned cyclepath", and alternative-191: "Build a ring around the city center" belong to issue-188, and have been proposed by

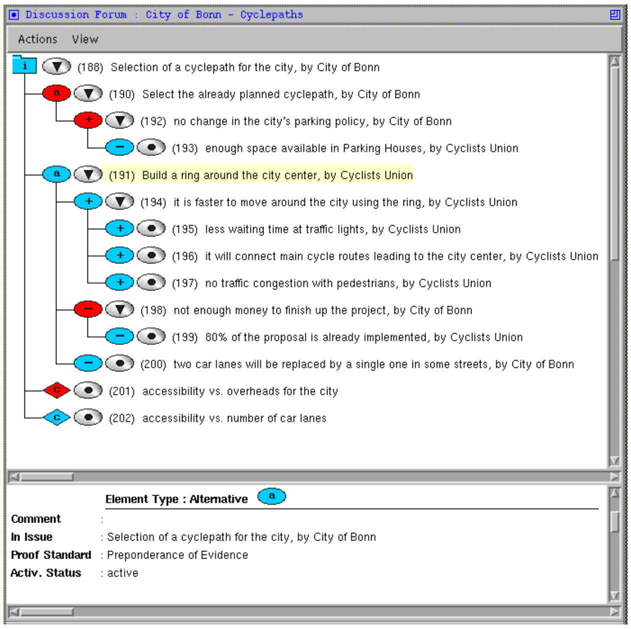

Figure 1: The system's basic user interface.

the City of Bonn and the Cyclists Union, respectively). Issues can be inside other issues in cases where some alternatives need to be grouped together. For instance, assume that there were two potential ring plans, say "Build a ring around the city center, according to the x plan" and "Build a ring around the city center, according to the y plan". In this case, the last two alternatives should have been grouped in an internal issue (say, the *subissue* "select the most appropriate plan for a city ring") and, after the related discourse, the selected (best) one would have been compared to alternative-190. Only one alternative in each issue is finally proposed by the system.

Positions are asserted in order to support the selection of a specific course of action (alternative), or avert the agents' interest from it by expressing some objection. For instance, position-192 "no change in the city's parking policy" has been asserted to support alternative-190, while position-198 "not enough money to finish up the project" to express the City's objection to alternative-191. Positions may also refer to some other position in order to provide additional information about it, e.g., position-193 "enough space available in Parking Houses" (arguing *against* position-192), and position-195 "less waiting time at traffic lights" (arguing *in favor* of position-194). A position always *refers to* a single other position or alternative, while an alternative is always *in* a single issue.

In decision making environments, one has usually to define priorities among actions and weigh different criteria. Unfortunately, well defined utility and probability functions regarding properties or attributes of alternatives (used in traditional approaches), as well as complete ordering of these properties are usually absent.

In HERMES, *constraints* provide a qualitative way to weigh reasons for and against the selection of a certain course of action. A constraint is a tuple of the form [position, preference_relation, position], where the preference relation can be *more (less) important than* or *of equal importance to*. Constraints may give various levels of importance to alternatives. Like the other argumentation elements, they are subject to discussion; therefore, they may be "linked" with positions supporting or challenging them. In Figure 1, constraint-201: "accessibility vs. overheads for the city" expresses the preference relation "position-194 is more important than position-192", while constraint-202: "accessibility vs. number of car lanes" the relation "position-194 is more important than position-200".

Two types of preferences can be expressed by the system: (i) *Local*, when a constraint refers to a position, or another constraint. In this case, the positions that constitute the constraint must refer to the same element (i.e., have the same father). In the example shown in Figure 1, a preference of this type can be expressed by the constraint "position-195 is less important than position-197". A constraint of the form "position-195 is less important than position-194" is not permitted. (ii) *Non-local*, when a constraint refers to an issue. In this case, its consistuent positions must refer to alternatives (not necessarily the same one) of this very issue (e.g., constraint-201 and constraint-202).

Proof Standards

Alternatives, positions and constraints have an *activation label* indicating their current status (it can be *active* or *inactive*). This label is calculated according to the argumentation underneath and the *type of evidence* specified for them. In general, different elements of the argumentation, even in the same debate, do not necessarily need the same type of evidence. Quoting the well-used legal domain example, the arguments required to indict someone need not be as convincing as those needed to convict him (Farley and Freeman 1995). Therefore, a generic argumentation system requires different *proof standards* (work on AI and Law uses the term *burdens of proof*). In the sequel, we describe the ones implemented in our system (the names have the same interpretation as in the legal domain). We do not claim that the list is exhaustive; other standards, that match specific application needs, can be easily incorporated to the system.

Scintilla of Evidence (SoE): according to this proof standard, a position p_i is active, if at least one active position argues in favor of it: $active(p_i) \Leftrightarrow \exists\, p_j\,(active(p_j) \wedge in\_favor(p_j, p_i))$.

Beyond Reasonable Doubt (BRD): according to BRD, a position p_i is active if there are not any active positions that speak against it: $active(p_i) \Leftrightarrow \neg\exists\, p_j\,(active(p_j) \wedge against(p_j, p_i))$.

Activation in our system is a recursive procedure; a change of the activation label of an element (alternative, position or constraint) is propagated upwards. When an alternative is affected during the discussion, the issue it belongs to should be updated since a new choice may be made. This brings us to the last proof standard, namely *Preponderance of Evidence* (PoE), used for the selection of the best alternative (however, it can also be used for activation/inactivation of positions). In the case of PoE, each position has a $weight = (max\_weight + min\_weight)/2$, while an alternative has $weight = min\_weight$. Max_weight and min_weight are initialized to some pre-defined values (in the following, we assume that initially $min\_weight = 0$ and $max\_weight = 10$; this may be changed by preference constraints). The *score* of an element e_i is used to compute its *activation label* (score is calculated on-the-fly). If an element does not have any arguments, its score is equal to its weight; otherwise, the score is calculated from the weights of the active positions that refer to it:

$$score(e_i) = \sum_{\substack{in\_favor(p_j, e_i) \wedge \\ active(p_j)}} weight(p_j) - \sum_{\substack{against(p_k, e_i) \wedge \\ active(p_k)}} weight(p_k)$$

Preponderance of Evidence (PoE): According to this standard, a position is *active* when the active positions that support it outweigh those that speak against it: $active(p_i) \Leftrightarrow score(p_i) \geq 0$. Concerning alternatives, PoE will produce positive activation label for a_i when there are no alternatives with larger score in the same issue: $active(a_i) \Leftrightarrow \forall\, a_j\ in\_issue(a_i), (score(a_j) \leq score(a_i))$.

In the discussion instance of Figure 1, the proof standard is SoE for all positions and PoE for alternatives. Position-192 and position-198 are inactive (their accompanying icons are shown in dark grey color) since position-193 and position-199, respectively, are active and speak against them. On the contrary, position-194 is active (the accompanying icon appears in light grey color) since there is at least one active position that speaks in favor of it (actually, there are three such positions; even if two of them become inactive at a future instance of the discussion, position-194 will remain active). Active positions are considered "accepted" due to discussion underneath (e.g., strong supporting arguments, no counter-arguments), while inactive positions are (temporarily) "rejected".

Similarly, active alternatives correspond to "recommended" choices, i.e., choices that are the strongest among the alternatives in their issue. Note that alternative-190 is not actually supported or objected by any position (only the inactive position-192 is underneath), while alternative-191 is actually supported by position-194 and objected by position-200 (position-198 is inactive). The mechanisms for the calculation of activation labels of alternatives always depend on the related constraints and will be discussed in the following subsection.

The activation label of constraints is decided by two factors: the discussion underneath (similarly to what happens with positions) and the activation label of their consistuent positions. In Figure 1, constraint-201 is currently inactive because position-192 is also inactive. According to the evolution of the discussion, the insertion of position-193 inactivated position-192, which in turn inactivated constraint-201. Both constraints have BRD as proof standard, therefore constraint-202 is active (no active positions speak against it). If in the sequel of the discussion, a new position inactivates position-193, this will result in a new activation of both position-192 and constraint-201 (assuming that nothing else related to them changes).

Consistency checking

Apart from an activation label, each constraint has a *consistency label* which can be *consistent* or *inconsistent*. Every time a constraint is inserted in the discussion graph, the system checks if both positions of the new constraint exist in another, previously inserted, constraint. In the affirmative case, the new constraint is considered either *redundant*, if it also has the same preference relation, or *conflicting*, otherwise. A redundant constraint is ignored, while a conflicting one is grouped together with the previously inserted constraint in an issue automatically created by the system, the rationale being to gather together conflicting constraints and stimulate further argumentation on them until only one becomes active.

If both positions of the new constraint do not exist in a previously inserted constraint, its consistency is checked against previous active and consistent constraints referring to the same element (or belonging to the same issue). This process is based on a polynomial ($O(N^3)$, N the number of the associated positions) *path consistency* algorithm. Although path consistency, as most discourse acts described in the sequel, interacts with the database where the discussion graph is stored (*Oracle 7* is used), the algorithm is very efficient. Even for non-local preferences involving issues with numerous alternatives and positions linked to them, execution time is negligible compared to communication delay.

Active and consistent constraints participate in the weighting scheme (only constraint-202 in the example of Figure 1). In order to demonstrate how the algorithm for altering weights works, we use the example of Figure 2. There exist five positions and four constraints that relate them as illustrated in Figure 2a. The arrowed lines correspond to the "more important than (>)" relation (e.g., $p_1 > p_2$) and the dotted line to the "equally important to (=)" relation (e.g., $p_3 = p_4$). First, path consistency explicates all ">" relations (Figure 2b). Then, *topological sort* is applied twice to compute the possible maximum and minimum weights for each position (Figure 2c). The *weight* is the average of the new *max_weight* and *min_weight*: $weight(p_1) = 6$, $weight(p_2) = 4.5$, $weight(p_3) = 5$, $weight(p_4) = 5$ and $weight(p_5) = 4$.

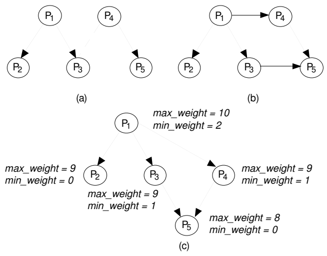

Figure 2: The weighting scheme.

The basic idea behind the above scheme is that the weight of a position is increased every time the position is more important than another one (and decreased when is less important), the aim being to extract a total order of alternatives. Since only partial information may be given, the choice of the initial maximum and minimum weights may affect the system's recommendation. Furthermore, this weighting scheme is not the only solution; alternative schemes, based on different algorithms, are under implementation.

The scores of alternative-190 and alternative-191 in Figure 1 are 0 and 1, respectively, since the former has no active positions "linked" to it, while for the latter, position-194 and position-200 have scores 5.5 and 4.5, respectively, and position-198 is inactive. Therefore, alternative-191 is active and recommended by the system. If position-192 were active, the scores of alternative-190 and alternative-191 would be 4.5 and 1, respectively. In this case, alternative-190 would be the recommended one.

Discourse Acts

Argumentation in our framework is performed through a variety of *discourse acts*. These acts may have different functions and roles in the argumentative discourse. We classify them in two major categories: *agent acts* and *internal (system) acts*. Agent acts concern user actions and correspond to functions directly supported by the user interface. Such functions include the opening of an issue, submission of an alternative, etc.

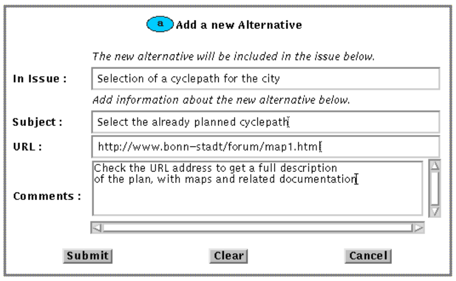

Figure 3: Adding a new alternative.

The applet window for adding a new alternative to an existing issue is shown in Figure 3. When an alternative alt_i is added to another alt_j (and not directly to the issue iss_j where alt_j belongs), a new issue iss_i is automatically created inside iss_j (a similar applet window is used in such a case). Both alt_i and alt_j are now put inside the new issue and compared through update(iss_i). Update(iss_j) will be called from update(iss_i) and the recommended choice between alt_i and alt_j will be compared against the other alternatives of the external (initial) issue. Note that in Figure 3, users can give a *subject* (title) of the new alternative, but also provide more details about their entry through the URL pane. In such a way they can "attach" to their elements pieces of text, images, etc.

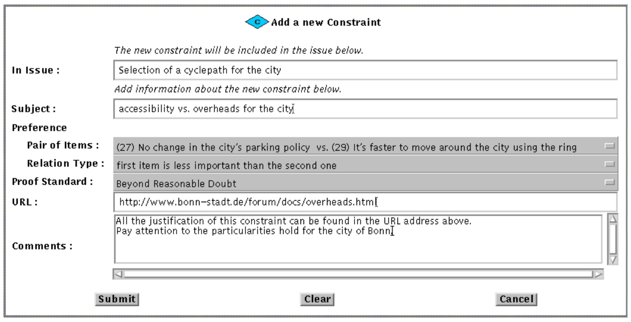

Figure 4: Adding a new constraint.

Figure 4 illustrates the applet window for adding a new constraint to an issue. Depending on the argumentation element selected for its insertion, the *pair of items* pane provides users a menu with all valid pairs, preventing users from making errors in expressing a preference. The *relation type* menu includes the preference relations *more (less) important than* and *equally important to*. The pseudo-code for adding a constraint to a position is as follows:

```
add_Constr._to_Posit.(constraint con, position pos) {
    if (redundant(con)) return; /* ignore */
    refersTo(con)=pos;
    for all constraints con_j that refer to pos
        if (conflicting(con_j, con))
        {       create new issue iss;
                In(iss)=con_j; In(iss)=con;
                update(iss); return; }
    if (contains_inactive(con)) ¬active(con);
        else { active(con);
                if (¬consistent_with_previous(con))
                        ¬consistent(con);
                else {   consistent(con); update(pos); }}}
```

Redundant and conflicting constraints were discussed earlier. Contains_inactive(con) checks whether one or both positions that con consists of are *inactive*, in which case con becomes *inactive*. The justification is that it does not make sense to argue about the relevant importance of positions that are rejected (con will be activated automatically when the compared positions get activated - see activate(*position pos*)). If con does not contain_inactive, then it is checked for consistency. Only if it is found consistent, the position that it refers to is updated, since inconsistent constraints do not participate in the weighting scheme.

Internal acts are functions performed by the system in order to check consistency, update the discussion status and recommend solutions. These functions are called by the agent acts and are hidden from the end user. For instance, consistent_with_previous(con), called by add_Constr._to_Posit., constructs a graph similar to Figure 2 and applies path consistency. Other representative internal acts are:

```
boolean compute_activation(position pos) {
boolean ¬ status_changed, old_activation=activation(pos);
switch Proof_Standard(pos) {
   case Scintilla of Evidence {
          ¬activation(pos);
          for all positions pos_j that refer to pos
                 if (active(pos_j) ∧ in_favor(pos_j, pos))
                 {       activation(pos); break; }}
   case Beyond Reasonable Doubt {
          activation(pos);
          for all positions pos_j that refer to pos
                 if (active(pos_j) ∧ against(pos_j, pos))
                 {       ¬activation(pos); break; }}
   case Preponderance of Evidence {
          score(pos)=0;
          calculate_weights(pos);
          for all positions pos_j that refer to pos
                 if (active(pos_j) ∧ in_favor(pos_j, pos))
                        score(pos) += weight(pos_j);
                 else if (active(pos_j)∧ against(pos_j, pos))
                        score(pos) -= weight(pos_j);
          if (score(pos) ≥ 0) activation(pos);
                 else ¬activation(pos) }}
if (old_activation ! = activation(pos)) status_changed;
return status_changed; }
```

`Compute_activation` returns *status_changed*, which is *true* if the activation label of the position changed. Activation is calculated according to the proof standard used. Proof standards are a straightforward implementation of the previously given definitions. In case of PoE, calculate_weights(*pos*) calls topological sort to compute the weights of the positions in favor and against *pos*.

```
update(position pos) {
if (compute_activation(pos)) {
    if (active(pos)) activate(pos);
        else inactivate(pos);
    update(RefersTo(pos)); }}    /*propagate upwards */
```

`Update` calls `compute_activation` to check whether activation label has changed. If this is the case, `activate` (see below) or `inactivate` are called and the change is propagated upwards.

```
activate(position pos) {
pos_j=refersTo(pos);
for all constraints con_j that refer to pos_j
   if (¬active(con_j) ∧ con_j has a prefer. relation on pos)
          if (compute_activation(con_j)) {
                 if (consistent_with_previous(con_j)
                        consistent(con_j);
                 else ¬consistent(con_j); }}
```

Activation of a position *pos* involves the retrieval of the constraints where *pos* appears (these constraints must refer to the same position as *pos*) and a check as to whether they can now become *active* and *consistent*. Inactivation of positions is similar, in the sense that when a position becomes inactive, the system searches for the constraints where the position appears and inactivates them as well. This may cause some other inconsistent constraints to become consistent, so consistency check is performed again for the related constraints. A number of additional internal acts were implemented for the activation/inactivation of constraints, update of issues, etc. The procedures described in this section, although only a subset of the whole set of functions performed by the system, give an indication of its dynamic structure. A single insertion in the discussion graph may update a large portion of the tree. Every time there is a change, the status of the argumentation elements is recorded in the database.

Conclusion

In general, it is very difficult to completely automate the processes involved in argumentative discourse. Rather, HERMES acts as an *assistant* and *advisor*, by facilitating communication and recommending solutions, but leaving the final enforcement of decisions to the agents. The use of information technology may assist in various ways. One important goal is to provide easy access to the current knowledge, at any time. Another, and this is the primary goal of the project, is to provide direct computer support for argumentation in a decision making environment, where the quality and acceptability of decisions depends not only on the availability of accurate information, but also on the fairness and openness of the procedure.

References

Brewka, G. 1994. A Reconstruction of Rescher's Theory of Formal Disputation Based on Default Logic. In Working Notes of the AAAI'94 Workshop on Computational Dialectics, 15–27.

Conklin, J., and Begeman, M.L. 1987. gIBIS: A Hypertext Tool for Team Design Deliberation. In Proc. of Hypertext'89 Conference, 247–252. New York:ACM.

Farley, A.M., and Freeman, K. 1995. Burden of Proof in Legal Argumentation. In Proc. of the 5th Int. Conference on AI and Law, 156–164. New York:ACM.

Fischer, G., McCall, R., and Morch, A. 1989. JANUS: Integrating Hypertext with a Knowledge-based Design Environment. In Proc. of the Hypertext'89 Conference, 105–117. New York: ACM.

Geffner, H., and Pearl, J. 1992. Conditional Entailment: Bridging two Approaches to Default Reasoning. *Artificial Intelligence* 53(2-3):209–244.

Gordon, T. 1995. *The Pleadings Game: An Artificial Intelligence Model of Procedural Justice.* Kluwer.

Hurwitz, R., and Mallery, J.C. 1995. The Open Meeting: A Web-Based System for Conferencing and Collaboration. In Proc. of 4th Int. WWW Conference.

Kreamer, K.L., and King, J.L. 1988. Computer-based systems for cooperative work and group decision making. *ACM Computing surveys* 20(2): 115–146.

Rescher, N. 1977. *Dialectics: A Controversy-Oriented Approach to the Theory of Knowledge.* Albany: State University of New York Press.

Rittel, H., and Webber, M. 1973. Dilemmas in a General Theory of Planning. *Policy Sciences* 155–169.

Smolensky, P., Fox, B., King, R., and Lewis, C. 1987. Computer-aided Reasoned Discourse, or How to Argue with a Computer. In Guindon, R. (ed.) *Cognitive Science and its Applications for Human-Computer Interaction*, 109–162. Hillsdale, NJ: Erlbaum.

Streitz, N., Hannemann, J., and Thüring, M. 1989. From ideas and arguments to hyperdocuments: Travelling through activity spaces. In Proc. of the Hypertext'89 Conference, 343–364. New York: ACM.

Suthers, D., Weiner, A., Connelly, J., and Paolucci, M. 1995. Belvedere: Engaging Students in Critical Discussion of Science and Public Policy Issues. In Proc. of the 7th World Conference on AI in Education, 266–273.