

trees consist of rhetorical goals ordered by subgoal and importance (to the ideologue) relationships. These goals are used both to select historical events to include in the story and to “spin” the event in an ideologically-consistent manner. The rule-based natural language generator (NLG) generates the narrative text once specific facts have been selected and connected to make a story. The storyboard serves as a working memory for processes that impose a narrative order on event spins created by the goal tree. Rhetorical devices are connecting pieces of text with accompanying constraints on story structure. These devices are used to create narrative connections between historical events. Finally, the multimedia database contains the audio/visual elements for the assembled documentary. Once a narrative track has been constructed, information retrieval techniques are used to match the “best” indexed multimedia elements to the appropriate pieces of text. Once the multimedia elements have been selected, the resulting documentary is displayed, layering text-to-speech synthesis of the narrative track, and the video and audio elements.

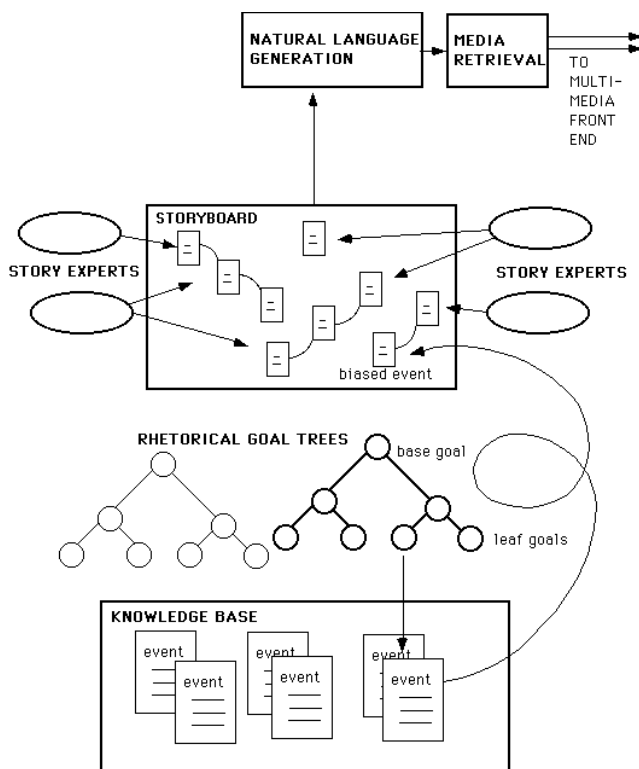


Figure 3: *Terminal Time* architecture

The audience's responses to the questions influence the machine by selecting and editing rhetorical goal trees, selecting a set of rhetorical devices, and placing constraints on the storyboard. In a sense, the audience's responses parameterize the machine. The responses activate structures and processes; the machine then autonomously generates a biased history.

The rest of this paper describes the artistic goals of

Terminal Time, provides more detail about each of the architectural components, and describes our experiences performing *Terminal Time*.

Artistic Goals

Documentary Form

The popular model of the documentary film in America today, most clearly exemplified by Ken Burns' "The Civil War," has the familiar structure of Western narrative. The rhetorical structure invariably involves a crisis situation, a climax, and a clear resolution. Generally there is one prevailing narrative, one interpretation of the historical facts presented. The documentary situates itself as an objective retelling of history, making it difficult for the viewer to question the authority of the presented viewpoint.

Terminal Time imitates the model of this “cookie-cutter documentary” with a machine that produces and reproduces it, until the model itself is revealed for the tool of ideological replication that it has become. *Terminal Time* generates endless variations of documentaries that have the “look and feel” of the traditional, authoritative PBS documentary. Yet these generated documentaries are clearly charged with a strong ideological bias derived from the audience's responses. In this way *Terminal Time* invites the viewer to question the implicit authority of documentary presentations of history.

Utopian Navigation

There is a great deal of industry hype surrounding interactive media and computing. Typically such experiences are promoted through a rhetoric of utopian navigation. According to such rhetoric, the computer provides unlimited access to information and experience, a pure source of empowerment that imposes no interpretation on the data that is processed. Other familiar tropes in this rhetoric include: Real-time, Immersion and Virtuality -- promising the thrill of reality or hyper-reality, without the effort, right from one's own PC. Microsoft's ads softly beguile us with the question “Where do you want to go today?”

With *Terminal Time*, we play with these notions by building a program that engages in active interpretation and construction of the interactive experience. While the resulting constructed histories clearly respond to audience input, the system has a mind of its own, pushing the story into extremes that the audience did not intend. Thus value-free navigation gives way to a value-laden interpretation. *Terminal Time* is a program that bites back.

Audience Experience

Utilizing indirect questionnaires as a user interface, the system essentially target markets each audience with an appropriate history. Rather than asking audiences what

type of history they would like, or how they would like to navigate through history, they are asked questions about their own demographics and psychographics: their work status, what cultural trends they find most disturbing, how well they get along with others, etc. The resulting history is like holding a funhouse mirror to the audience; it reflects an exaggerated and distorted view of the audience's biases.

As the history begins 1000 years ago, the audience should experience a comfortable sense of historical authority engendered by the familiar documentary form and the remoteness of the historical events. As the history unfolds, the effect of the periodic audience polls becomes more and more apparent. The increased bias evident in the history should begin creating a tension with regard to the veridicality of the history (a sense of "wait a minute, this doesn't seem quite right..."). Ideally, this tension should reach a maximum as the piece moves into modern history.

Knowledge Base

Upper Cyc Ontology

The knowledge base consists of higher order predicate statements about historical events, definitions of ontological entities used in the historical event descriptions (individuals and collections), and inference rules. *Terminal Time's* ontology is based on the Upper Cyc Ontology, the top 3000 most general terms in the Cyc ontology (Lenat 1995). The Upper Cyc Ontology is available free of charge from Cycorp¹. It does not include any other components of Cyc (theorem prover, natural language engine, database, etc.). However, the upper ontology provides a useful set of distinctions in terms of which the more specific ontology needed by *Terminal Time* can be defined.

Example Historical Event

Figure 4 shows the representation of The Giordano Bruno

```

;; Giordano Bruno
($isa %GiordanoBrunoStory %HistoricalEvent)
($isa %GiordanoBrunoStory %IdeaSystemCreationEvent)
($isa %GiordanoBrunoStory %Execution)
(%circa %GiordanoBrunoStory (%DateRangeFn
  (%CenturyFn 16) (%CenturyFn 17)))
($eventOccursAt %GiordanoBrunoStory $ContinentOfEurope)
($performedBy %GiordanoBrunoStory %GiordanoBruno)
($outputsCreated %GiordanoBrunoStory %GiordanoBrunosIdeas)
($isa %GiordanoBrunosIdeas $PropositionalInformationThing)
($isa %GiordanoBrunosIdeas $SomethingExisting)
(%conflictingMOs %GiordanoBrunosIdeas %MedievalChristianity)
($isa %GiordanoBrunosIdeas %IdeaSystem)
($performedByPart %GiordanoBrunoStory
  %TheRomanCatholicReligiousOrg)
($objectActedOn %GiordanoBrunoStory %GiordanoBruno)

```

Figure 4: Example knowledge base representation

¹ <http://www.cyc.com/>

story. Those terms preceded by a "\$" are defined in the Upper Cyc Ontology. Those terms preceded by "%" are defined within the TT ontology in terms of the Upper Cyc Ontology.

Figure 4, translated into English, states:

The Giordano Bruno story, a historical event occurring in the 16th and 17th century, involved the creation of a new idea system and an execution. The idea system created in this event conflicts with the idea system of medieval Christianity. Both Giordano Bruno and a portion of the Roman Catholic Church were the performers of this event. Giordano Bruno was acted on (he was executed) in this event.

In this particular representation of the story of Giordano Bruno, both the creation of his philosophical writings and his execution by the Roman Catholic Church are treated as a single compound event. If we wanted *Terminal Time* to be able to treat these two events separately (perhaps talking about Bruno's writings without mentioning his eventual execution), they could be represented as two sub-events. In general, events are only represented as deeply as is needed by the rhetorical goal trees, storyboard, and natural language generator.

Inference Engine

The inference engine, implemented in Common Lisp, is based on the interpreter implementing higher-order hereditary Harrop logic described in (Elliott and Pfenning 1991). Hereditary Harrop logic allows knowledge base entries (the program, thinking in logic programming terms) to consist of Horn clauses, and queries (goals) to consist of all the standard Prolog-like goals (atomic goals, conjunctions, disjunctions, existentials), plus embedded implications (assumptions). The interpreter also includes extra-logical support for operations such as unifying logic variables against a function evaluated by Lisp.

The inference engine is used to answer all queries about historical events. For example, in the discussion below of ideological goal trees, the historical event tests are all made using the inference engine.

Ideological Goal Trees

Terminal Time organizes ideological bias with goal trees, adapted from Politics (Carbonell 1979). In Politics, ideology is encoded as a set of goals held by the ideologue. The goals are organized via subgoal links (not corresponding exactly to either the conjunctive or disjunctive notion of subgoal) and relative importance links. The relative importance links place an importance partial order over the subgoals. For example, in Politics, the US Conservative ideologue's most important goal is *Communist Containment*. This goal has a number of subgoals such as *Have a Strong Military*, *Aid Anti-Communist Countries*, etc. Though *Have a Strong Military* and *Aid Anti-Communist Countries* are sibling subgoals, *Have a Strong Military* has a higher relative importance. In

addition to their own goal tree, an ideologue also possesses beliefs about the goal trees of others. In Carbonell's system, the goal trees were used to organize inferences made by a news story understanding system.

In *Terminal Time*, the goal tree has been modified to represent the goals of an ideological story-teller. Rather than having goals to modify the world, the story-teller has rhetorical goals to show that something is the case. For example, the *Anti-Religious Rationalist* possesses the goals show in Figure 5 during the first segment of the history. The indented goals are subgoals.

The leaf goals in the goal tree are used to organize two kinds of knowledge: a set of tests for recognizing when a historical event is potential fodder for satisfying the rhetorical goal, and a set of plans for actually constructing the description of the event to satisfy the goal (the event spin).

```
show-religion-is-bad
  show-religion-causes-war
  show-religion-causes-crazy-self-sacrifice
  show-religion-causes-oppression
  show-religion-causes-self-abuse
  show-thinkers-persecuted-by-religion
show-halting-rationalist-progress-against-religion
  show-thinkers-opposing-religious-thought
  show-thinkers-persecuted-by-religion
```

Figure 5: Anti-Religious Rationalist goal tree
Notice that the leaf goal *show-thinkers-persecuted-by-religion* is a subgoal of two higher level goals. Satisfying this goal satisfies both higher-level goals.

Tests for Event Applicability

An ideologue needs a way of recognizing when a historical event could be used to satisfy a goal (make an ideological point). For example, the *Anti-Religious Rationalist* must be able to recognize that the Giordano Bruno story can be used to show that thinkers are persecuted by religion. This involves recognizing that a religious organization does something negative to a thinker because of the thinker's thoughts. In the current version of *Terminal Time*, this test determines whether an event involves both the creation of

```
(%and
 ($isa ?event %IdeaSystemCreationEvent)
 ($isa ?event %Execution)
 ($outputsCreated ?event ?newIdeas)
 (%conflictingMOs ?newIdeas ?relBeliefSystem)
 ($isa ?relBeliefSystem $Religion))
```

Figure 6: Example event applicability test

an idea system and an execution, and whether the idea system conflicts with some religious belief system. The formal syntax of this test expressed in the language of the inference engine is shown in figure 6.

This test assumes that the execution must have been

performed by the religious organization in response to the creation of the conflicting idea system. Further, it assumes that the only forms of persecution are execution. These simplifying assumptions work because given the current content of the knowledge base, this applicability test is sufficient to locate events that can be slanted as forms of religious persecution. As new events involving religious persecution are added to the knowledge base, the test may have to be changed (most likely broadened).

Plans for Event-level Story Generation

Once an event has been recognized as applicable to a rhetorical goal of the ideologue, additional knowledge is necessary to spin the event in such a way as to satisfy the rhetorical goal. This knowledge is represented as rhetorical plans. These plans put a "spin" on the event (referred to as a *spin*) by selecting a subset of the event knowledge represented in the KB to place on the storyboard.

```
Describe the individual who called for the war,
mentioning their religious belief
Describe the religious goal of the war
Describe some event happening during the war
Describe the outcome
```

Figure 7: Rhetorical plan outline

Rhetorical plans are the mechanism by means of which a rhetorical goal can place its unique view on an event.

The plan language is similar to the rule language for natural language generation, except that the atomic actions for the NLG rule language emit strings while the atomic actions for the plan language add syntactic units to an event spin. See the section on NLG rules for a more detailed description of the logic allowed in rhetorical plans.

An outline of an example plan for *Show that religious thought leads to war* is shown in Figure 7.

In addition to the knowledge elements selected by the rhetorical plan, the name of the rhetorical goal and the names of all of its parents are added to the spin. The goal name(s) tell the rhetorical devices and natural language generator which goal(s) a particular spin is satisfying.

Rhetorical Devices

After the rhetorical goals are done producing event spins, the storyboard now contains an unordered collection of spins. Rhetorical devices connect spins together to form a story. Rhetorical devices consist of an English sentence(s) (actually represented as NLG rules) and accompanying logical tests that can be used to connect spins together. For example, the sentence "Yet progress doesn't always yield satisfaction" can be used to connect several spins describing the positive effects of technological progress and several spins describing social or environmental problems arising from technological progress. The associated tests require that all the spins preceding the rhetorical device must be positive technological, artistic, or

industrial progress, and that all the spins following the rhetorical device must be negative effects of progress.

Prescope and Postscope Tests

The prescope of a rhetorical device is the ordered collection of spins preceding the device. The postscope is the ordered collection of spins following the device. The prescope and postscope tests are constraints (interpreted by the inference engine) that the preceding and following spins must satisfy in order for the rhetorical device to be applicable (that is, able to glue the spins together). Scope tests can either require that all the spins in the scope satisfy the test or that at least one spin in the scope satisfies the test. In addition, the scope range and length can be specified. The scope length is the number of spins to include in the scope; the default is 1 (that is, only the preceding or following spin must satisfy the test). The scope range specifies the range of spins that can be searched for a satisfying scope; the default is (1 1) (the range consists of only the immediately preceding or following spin).

Rhetorical Device NLG Rule

Associated with each rhetorical device is an NLG rule for generating the English sentence associated with the device. For some rhetorical devices, this may be a simple rule generating a single canned sentence. For other (more flexible) rhetorical devices, the rule may be passed arguments (which were bound by the scope tests) which influence the generated sentence.

Example Rhetorical Device

An example rhetorical device is shown in figure 8.

```
(def-rhetdev :name :non-western-religious-faith
:prescope-length 2
:prescope-test (:all-events-satisfy (%and
($isa ?event %HistoricalSituation)
(:kb ($eventOccursAt ?event %FirstWorld))
(%rhet-goal :show-religion-is-good)))
:postscope-test (:some-event-satisfies ?spin (%and
($isa ?event %HistoricalSituation)
(:kb ($eventOccursAt ?event %NonFirstWorld))
(%rhet-goal :show-religion-is-good)))
:nlg-rule :generate
:nlg-context-path (:non-western-religious-faith))
```

Figure 8: Example rhetorical device

This rhetorical device, employed by the *Pro-religious Supporter*, is used to connect a couple of spins describing Western religious faith, with an example of non-Western religious faith. The prescope-length is 2; since no prescope-range is specified, it defaults to the preceding two events. Thus the prescope test must be satisfied by the preceding two spins and only the immediately preceding two spins. The test requires that the event occurred in the First World (represented in the ontology as a collection of

geographical regions which includes regions such as Europe) and that it satisfied the rhetorical goal *show religion is good*. The *%rhet-goal* term was added to the event spin during rhetorical plan processing (when the rhetorical goal sticks the spin on the blackboard). Most rhetorical devices test the satisfied rhetorical goal in their scope tests; these goal labels indicate how an event has been slanted in order to create a specific even spin. The postscope test similarly tests whether the immediately following event spin satisfies the goal *show-religion-is-good* in the *non-First World*. In the event that the constraints are satisfied, text will be generated by the NLG rule.

Story Generation

Once a collection of event spins has been placed on the storyboard, a historical story can be generated. For each of the three periods of the documentary, each ideologue has a list of storyboard constraints. The storyboard constraint for section 1 of the *Anti-Religious Rationalist* is shown in figure 9.

```
(%rhet-goal :show-religion-is-bad)
(%rhet-goal :show-religion-is-bad)
(%rhet-goal :show-religion-is-bad)
(%rhet-goal :show-religion-is-bad)
(%rhet-goal :show-halting-rationalist-progress)
(%and (%rhet-goal :show-halting-rationalist-progress)
(%rhet-goal :show-religion-is-bad))
```

Figure 9: *Anti-Religious Rationalist* storyboard constraint

The length of the constraint list determines how many event spins will be included in the story section. In this example, six spins will be included. Each test in the list constrains the spins that can appear in each position in the story. Typically these are constraints on the rhetorical goals that were satisfied to create the spin. In addition to the storyboard constraints, there is also an implicit temporal constraint that requires that spins appear in roughly chronological order.

To generate a story, the space of all sequences of event spins satisfying the storyboard constraints is searched for a sequence that can be satisfied by the current set of rhetorical devices. A sequence is satisfied if a rhetorical device with satisfied scope tests can be placed between every spin in the sequence. The resulting sequence of interleaved spins and rhetorical devices is a story.

NLG Rules

The NLG system generates English text for both event spins and rhetorical devices. NLG is accomplished using rules which map pieces of knowledge representation onto English. There is no deep theory of lexical choice or text planning. The goal of the NLG system is to produce high quality text for the stories generated on the storyboard. The rule language provides a framework in which a human author can write text ranging in granularity from canned

paragraphs down to individual words and phrases and provide the logic to map these varying chunks onto pieces of knowledge representation.

NLG Rule Syntax

Figure 10 provides an abstract example of a rule. This example makes use of most of the features supported by the rule language. This language is similar to the language used for rhetorical plans. All tests mentioned in the NLG rules are interpreted by the inference engine.

```
(def-nlrule
 :name :rule-name
 :context :some-context
 :test test over the rule arguments
 :body (:seq
        (:terminal
         (:string "string 1")
         (:keywords k1 k2 k3))
        (:cond
         ((test1 over rule arguments)
          (:terminal...))
         ((test 2 over rule arguments)
          (:bag-one-of
           step1...
           stepn)))
        (:rule subrule args (context1 ... contextn))
        (:opt (:if (another test)
                  (:seq...))))))
```

Figure 10: NLG rule syntax

An NLG rule is identified by a name and a rule context. The rule context provides a name space in which NLG rule names are recognized. Each context defines a set of rules which are specialists in handling some particular NLG task. Typically, a separate rule context is used for each historical event found in the knowledge base. When generation is initiated, a rule name, arguments (list of knowledge representation elements for which English should be generated) and a context list are given. The context list provides a set of contexts (in order from most specific to most general) in which to search for rules matching the rule name.

When a rule with matching rule name is found, the test is evaluated against the arguments to determine whether to use that instance of the rule for generation. Within a context, there may be multiple rules with the same name (corresponding to different ways to accomplish the same generation task); the test is used to determine which of these rules should be applied given specific arguments.

Once a rule is found whose test evaluates to true, the rule body is interpreted. In the example rule, the rule body is a sequence of steps. Terminals are the atomic steps that emit language. In addition to emitting an English string, terminals emit keywords associated with the English string. These keywords are used by the multimedia retrieval subsystem to associate a video clip with the sentence.

The conditional step (*:cond*) allows generation to branch depending on tests over the rule arguments. The branches may either be individual terminals or an entire rule body. If none of the tests in a conditional succeeds, then the rule fails; the system will try to find other applicable rules for generation. The *bag-one-of* body in the second branch of the conditional chooses one of the steps at random to execute. This can be used to add some random variation to generation.

A rule may contain a call to another generation rule.

The *:opt* construct allows the insertion of an optional conditional step. If the conditional is satisfied, the conditional branch is executed. If the conditional fails, execution of the rule body continues after the optional step.

Video Sequencing

After natural language generation, the event spins and rhetorical devices have been rendered as English text. Video clips from the database of keyword-annotated clips must be sequenced against the text (which forms the narrative track) to create the complete documentary. Video sequencing takes place in two steps. First, the keywords associated with each sentence (the keywords emitted by terminals in NLG rules) are used to retrieve keyword annotated video clips using TF/IDF term-based retrieval (Salton and Buckley 1988). This creates a list of top-scoring video clips for each sentence (typically the top 4 or 5 are taken). Then a greedy forward and backward search through the narrative track is performed to try and maximize clip continuity while minimizing clip reuse. If a pair of consecutive sentences shares clips among their top scoring clips, this greedy search will play the top-scoring shared clip across both sentences.

Current Status

Currently *Terminal Time* contains 134 historical events and 1568 knowledge base assertions (in addition to the assertions in the Upper Cyc Ontology). Nine major ideologues are represented using a total of 222 rhetorical goals, 281 rhetorical devices, and 578 NLG rules. The video database contains 352 annotated 30 second clips. *Terminal Time* has been performed in 14 venues, including the Carnegie Museum of Art, the Warhol Museum, and as the Walker Museum's entry in Sonic Circuits. Work continues in adding new events, goals, devices, NLG rules and video clips.

Performance Experiences

One way to evaluate an AI-based interactive art work is to evaluate the audience response to the system, to examine whether the AI architecture supports an audience interaction which successfully conveys the artistic intentions of the piece. Our knowledge of the audience reaction to *Terminal Time* comes both from observing

audiences during a performance and from the audience discussion period we always hold after a performance.

During performances, the audience is highly engaged with the piece. During the interactive polls, segments of the audience sometimes compete for control, clapping and shouting to make their choice the winner. At other times, the audience laughs when a choice meets with silence (no one wants to vote for it). Sometimes the applause grows into a groundswell of whistling and clapping as it becomes clear that certain choices are nearly unanimous. As the audience watches the constructed histories, there is often laughter, and sometimes groans and gasps. These reactions tend to grow as the documentary proceeds, indicating that the ideological bias is indeed becoming stronger and more visible as the history proceeds.

The discussion period tends to be quite animated, with the audience offering many questions and comments. Common topics of discussion include the role of ideology in the construction of history, the nature of certain specific biases, and the experience of being unable to completely control over the machine. From both the audience reactions during the performance and the nature of the post-performance discussion period, *Terminal Time* is successfully creating an engaging audience experience in accord with our artistic intentions.

Related Work

Two of the earliest computational models of ideology are Abelson's Goldwater Machine (Abelson and Carroll 1965) and Carbonell's Politics (Carbonell 1979). The Goldwater Machine mimics the responses of conservative presidential candidate Barry Goldwater to questions about the Cold War. The Goldwater Machine's responses were driven by a Cold War masterscript describing typical roles and event sequences during the Cold War. Politics represents rhetorical goals in order to guide biased understanding of news stories. In contrast to both the Goldwater Machine and Politics, *Terminal Time* generates biased historical stories composed of multiple events, rather than answering individual questions.

Pauline (Hovy 1987) generates natural language text for news events subject to the pragmatic constraints of rhetorical goals. Rhetorical goals include goals of opinion (e.g. show that our side has good goals or takes good actions) and goals of style (level of formality, level of simplicity). Pauline knows about 3 events, but is able to produce 100 different descriptions of an event. Where Pauline has a deep architecture for generating descriptions of individual events, *Terminal Time* selects and connects multiple events to satisfy an ideological position.

Spinductor (Sack 2000) uses statistical techniques to classify bias in news stories. This system determines the ideological point-of-view expressed in stories about the Nicaraguan Contras. While the use of statistical techniques makes Spinductor robust, it is concerned with classification where *Terminal Time* is concerned with generation.

Some computer based documentaries support the user in

navigating through documentary materials (e.g. Davenport and Murtaugh 1995, Schiffer 1999). As a user interacts with the system, implicit queries retrieve and play annotated video clips. Where these systems support a user in exploring a documentary space through immediate navigation, *Terminal Time* autonomously generates biased documentaries in response to punctuated audience feedback.

Finally, *Terminal Time* differs from all these systems in self-consciously being a work of art. *Terminal Time* is a piece of interactive performance art designed to create an experience for an audience.

Conclusion

This paper has described *Terminal Time*, an AI-based interactive artwork which produces ideologically-biased documentary histories in response to audience feedback. A novel AI architecture developed for *Terminal Time* was described. Performance experience suggests that the architecture supports an audience interaction in accord with our artistic goals.

References

- Abelson, R., Carroll, J. 1965. Computer Simulation of Individual Belief Systems. *American Behavioral Scientist*, 8, 24-30.
- Carbonell, J. 1979. Subjective understanding: Computer models of belief systems. Ph.D. Thesis, Computer Science Department, Yale University, Research Report #150.
- Davenport, G., Murtaugh, M. 1995. ConText: Towards the Evolving Documentary. ACM Multimedia '95, November.
- Elliott and Pfenning. 1991. A semi-functional implementation of a higher-order logic programming language. In Peter Lee, editor, *Topics in Advanced Language Implementation*, pages 289-325. MIT Press.
- Hovy. 1987. Generating Natural Language Under Pragmatic Constraints. Ph.D. Thesis, Computer Science Department, Yale University, Research Report #521.
- Lenat. 1995. Cyc: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38, no. 11, November.
- Sack, W. 2000. "Actor-Role Analysis: Ideology, Point of View and the News," in *Narrative Perspectives: Cognition and Emotion*, Chatman S., Van Peer, W. (editors), New York: SUNY Press.
- Salton, G., Buckley C. 1988. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, Vol. 24 No. 5, pp. 513-523.
- Schiffer. S. 1999. The Rise and Fall of Black Velvet Flag: An "Intelligent" System for Youth Culture Documentary. *AAAI Fall Symposium on Narrative Intelligence*.