

# Expressive AI: A Semiotic Analysis of Machinic Affordances

Michael Mateas

Georgia Institute of Technology

Literature, Communication and Culture & The College of Computing

686 Cherry Street

Atlanta GA 30032-0615

michael.mateas@lcc.gatech.edu

## ABSTRACT

Expressive AI is a hybrid practice, combining artificial intelligence (AI) research and art making, that simultaneously focuses on the negotiation of meaning mediated by an art object *and* the internal structure of AI systems. These two apparently disparate views are unified through the concept of affordance: negotiation of meaning is conditioned by interpretive affordances while the internal structure of the AI system is conditioned by authorial affordances. This paper employs a structuralist semiotic analysis to unpack the notion of interpretive and authorial affordance, exploring the deep relationships between AI code structures, authorial intentionality, and culturally negotiated meaning.

## Keywords

Artificial intelligence, semiotics, art

## 1. Introduction

Art and artificial intelligence (AI) research appear to be quite different practices. Where art practice focuses on the negotiation of meaning as mediated by the art object, AI research focuses on internal system structure and the interaction between system and environment. My work in AI-based art and entertainment simultaneously engages in AI research and art making, a research agenda and art practice I call *Expressive AI* [10, 11].

Expressive AI has two major, interrelated thrusts: (1) *exploring the expressive possibilities of AI architectures* – posing and answering AI research questions that wouldn't be raised unless doing AI research in the context of art practice, and (2) *pushing the boundaries of the conceivable and possible in art* – creating artwork that would be impossible to conceive of or build unless making art in the context of an AI research practice.

Expressive AI is thus a hybrid practice simultaneously focusing on the negotiation of meaning *and* the internal structure of AI systems. These two apparently disparate views are unified through the concept of affordance: negotiation of meaning is conditioned by interpretive affordances while the internal structure of the AI system is conditioned by authorial affordances. In [11] I described how a focus on authorial expression changes the AI research

First published at COSIGN-2003,  
09 – 12 September 2003, University of Teesside (UK),  
School of Computing and Mathematics, Virtual Environments  
Group

agenda, positioned Expressive AI relative to both symbolic and embodied AI, and introduced the idea of interpretive and authorial affordance. This paper employs a structuralist semiotic analysis to unpack the notion of interpretive and authorial affordance, exploring the deep relationships between AI code structures, authorial intentionality, and culturally negotiated meaning.

## 2. Example Systems

This section provides brief descriptions of three AI-based artworks. These systems are used as examples throughout the rest of the paper.

### 2.1 Office Plant #1

Walk into a typical, high tech office environment, and, among the snaking network wires, glowing monitors, and clicking keyboards, you are likely to see a plant. In this cyborg environment, the silent presence of the plant fills an emotional niche. Unfortunately, this plant is often dying; it is not adapted to the fluorescent lighting, lack of water, and climate controlled air of the office. *Office Plant #1* [5] is an exploration of a technological object, adapted to the office ecology, that fills the same social and emotional niche as a plant. *Office Plant #1* employs text classification techniques to monitor its owner's email activity. Its robotic body, reminiscent of a plant in form, responds in slow, rhythmic movements to express a mood generated by the monitored activity. In addition, low, quiet, ambient sound is generated; the combination of slow movement and ambient sound thus produces a sense of presence, responsive to the changing activity of the office environment.



Figure 1. Office Plant #1.

*Office Plant #1* classifies incoming email into social and emotional categories using AI statistical text classification techniques. Given the categories detected by the email classifiers, a Fuzzy Cognitive Map (FCM) determines which behavior the plant should perform. The FCM is a neural network-like structure

in which nodes, corresponding to behaviors, are connected to each other by negative and positive feedback loops.

## 2.2 Terminal Time

*Terminal Time* [14] is a story generation system that constructs ideologically-biased documentary histories, consisting of spoken narrative, video sequence and sound track, in response to audience feedback measured by an applause meter. One of the goals of *Terminal Time* is to build a caricature model of the documentary film production process. Rather than “objectively” reporting a sequence of events through the eye of a camera (the implied production process in documentary film), events are instead selected and biased so as to satisfy an ideological position, assembled into a desired narrative, and only then is video footage selected to illustrate the constructed narrative. As a large-audience interactive artwork, *Terminal Time* allows an audience to explore the role of ideological bias in the construction of history. As an AI research system, *Terminal Time* integrates a novel model of ideologically-biased reasoning within a story-generation framework.

The architecture makes use of several representations and knowledge sources including: a knowledge base of historical events represented in an ontology based on the Upper Cyc Ontology, ideology-specific representations of rhetorical goals that select and “spin” events, rhetorical devices that can be used to “glue” spins together to form historical narratives, a plan-based natural language generator, and a database of term-indexed video clips.

## 2.3 Façade

*Façade* is an artificial intelligence-based art/research experiment in electronic narrative – an attempt to move beyond traditional branching or hyper-linked narrative to create a fully-realized, one-act interactive drama [12, 13]. *Façade* incorporates the player’s interaction with autonomous characters into a well-shaped dramatic arc with a clear inciting incident, progressive complication leading to a climax, and closure. In *Façade*, you, the player, play the character of a longtime friend of Grace and Trip, an attractive and materially successful couple in their early thirties. During an evening get-together at their apartment that quickly turns ugly, you become entangled in the high-conflict dissolution of Grace and Trip’s marriage.

Architecturally, *Façade* consists of a number of components. ABL (A Behavior Language) is a novel reactive planning language for authoring believable agents. ABL provides language support for authoring coordinated, multi-character dramatic action. The drama manager operationalizes dramatic beats. In dramatic writing, a beat is the smallest unit of dramatic value change, where dramatic values are properties of individuals or relationships such as trust, love, hope, etc. In *Façade* beats are architectural entities, consisting of preconditions, a description of the values changed by the beat, success and failure conditions, and joint behaviors (written in ABL) that coordinate the characters in order to carry out the specific beat. The drama manager attempts to sequence beats so as to incorporate player interaction while making specific dramatic arcs (value change graphs) happen. The natural language processing system employs semantic parsing to map dialog typed by the player into discourse acts (e.g. agree, disagree) and interprets the resulting discourse acts as a function of the current discourse context (most often

defined by the currently active beat). Finally, a custom non-photorealistic animation engine presents the story world as a real-time, 3D space through which the player can move, gesture, interact with objects, and talk with characters (dialog input is accomplished through typing).

## 3. Affordances

The notion of affordance was first suggested by Gibson [8] in his theory of perception and was later re-articulated by Norman [17] in the field of interface design. For Gibson, affordances are *objective*, actionable properties of objects in the world. For an animal to make use of the affordance, it must of course perceive it in some way, but for Gibson, the affordance is there whether the animal perceives it or not; an unperceived affordance is waiting to be discovered. For Norman, affordances become *perceived and culturally dependent*. That is, rather than viewing the relationship between sensory object and action as an independent property of the object+animal system, this relationship is contingent, dependent on the experiences of the perceiver within some cultural framework. For example, for a person who has spent the last 10 years using the web, blue underlined text now affords an action, clicking with a pointing device, with the expectation that this clicking will “follow a link” to another information node. If blue underlined text is used in a different interface merely as a way to emphasize text, this is likely to generate confusion because the hypothetical interface is violating an affordance. It is this second notion of contingent affordance that I use here. But note that though affordances are contingent, they are not arbitrary – affordances are conditioned by the details of human physiology (what we can sense, how our bodies move), by cultural memory, and by the perceivable physical properties of objects. While new affordances can come into existence, as illustrated by the link-following affordance of blue underlined text, these innovations are conditioned by earlier affordances (e.g. the physical affordances of computer mice) and take active cultural work to establish.

### 3.1 Interpretive Affordance

Interpretive affordances support the interpretations an audience makes about the operations of an AI system, conditioning the meanings negotiated between artist and audience. Interpretive affordances provide resources both for narrating the operation of the system, and additionally, in the case of an *interactive* system, for supporting intentions for action.

For AI-based art, narrative affordances support the audience in creating a story about the operation of the piece and how this operation relates to the artist’s intention. For example, imagine having *Office Plant #1* on your desk. The name, plus the physical form, prepares one to view the sculpture as a plant – it has identifiable parts that metaphorically relate to the stem, flower, and leaf of biological plants. The wooden box of the base, hammered finish of the flower, and whimsical piano-wire fronds topped with crinkled, copper-foil-wrapped spheres, give the plant a non-designerly, hand-built look that communicates that it is neither a consumer electronic toy nor serves any functional purpose. Yet it is clearly a machine – it hums quietly while operating, moves very slowly (the motion is visible only if you watch patiently), and, when returning to the desk after an absence, is sometimes in a different configuration than it was left in. The plant starts moving when email is received; over time one can

notice a correlation between the plant's physical poses and the email received. All of the perceived features of the plant, the materials used and the details of fabrication, the physical form, the temporal behavior, the relationship between this behavior and email, constitute the narrative affordances, the "hooks" that the plant's owner uses to make sense of the plant, to understand the plant in relationship to themselves and their daily activity.

For interactive art, intentional affordances support the goals an audience can form with respect to the artwork. The audience should be able to take an action and understand how the artwork is responding to this action. This doesn't mean that the artwork must provide simple one-to-one responses to the audience's actions. Such simple one-to-one responses would be uninteresting; rather, the poetics of the piece will most likely avoid commonly used tropes while exploring ambiguities, surprise, and mystery. But the audience should be able to understand that the system is responding to them, even if the response is unexpected or ambiguous. The audience should be able to tell some kind of unfolding story about their interaction with the work. Both the extremes of simple stereotyped responses to audience interaction making use of well-known tropes, and opaque incoherence with no determinable relationship between interaction and the response of the art work, should be avoided.

A concern with interpretive affordances is often alien to AI research practice. Though the role of interpretation is sometimes discussed (e.g. the Turing test is fundamentally about interpretation [20], Newell's knowledge level is an attribution made from *outside* an AI system [15]), most often AI systems are discussed in terms of *intrinsic* properties. But for artists, a concern with interpretive affordance is quite familiar; negotiating meaning between artist and audience is central to artistic practice. Expressive AI adopts this concern within the context of AI-based art. But Expressive AI also adopts a concern for the internal functioning of the artifact from AI research practice.

### 3.2 Authorial Affordance

The authorial affordances of an AI architecture are the "hooks" that an architecture provides for an artist to inscribe their authorial intention in the machine. Different AI architectures provide different relationships between authorial control and the combinatorial possibilities offered by computation. Expressive AI engages in a sustained inquiry into these authorial affordances, crafting specific architectures that afford appropriate authorial control for specific artworks.

This concern with the machine itself will be familiar to AI research practitioners. However, AI research practice often downplays the role of human authorship, focusing on the properties of the architecture itself independent of any "content" authored *within* the architecture. Multiple architectures are most often compared in a content-free manner, comparing them along dimensions and constraints established by theories of mind, or theories of brain function (not necessarily at the lowest, neuron level), or comparing their performance on established benchmark problems. For Expressive AI, the concern is with how the internal structure of the machine mediates between authorship and the runtime performance.

A focus on the internals of the machine itself is often alien to current electronic media practice; the internal structure of the machine is generally marginalized. The machine itself is

considered a hack, an accidental byproduct of the artist's engagement with the concept of the piece.

One might generalize in this way (with apologies to both groups): artists will kluge together any kind of mess of technology behind the scenes because the coherence of the experience of the user is their first priority. Scientists wish for formal elegance at an abstract level and do not emphasize, or do not have the training to be conscious of inconsistencies in, the representational schemes of the interface. [18]

In discussions of electronic media work, the internal structure of the machine is almost systematically effaced. When the structure is discussed, it is usually described at only the highest-level, using hype-ridden terminology and wishful component naming (e.g. "meaning generator", "emotion detector"). At its best, such discursive practice is a spoof of similar practice within AI research, and may also provide part of the context within which the artist wishes her work to be interpreted. At its worst, such practice is a form of obfuscation, perhaps masking a gap between intention and accomplishment, the fact that the machine does not actually do what is indicated in the concept of the piece.

Yet it is nonetheless the case that an artist's concern with the coherence of the audience experience, with the crafting of interpretive affordances, is entirely appropriate – creating an audience experience is one of the primary reasons the artwork is being made in the first place. So why should an artist concern herself with authorial affordances, with the structural properties of the machine itself? Because such a concern allows an artist to explore expressive possibilities that can only be opened by a simultaneous inquiry into interpretive affordance and the structural possibilities of the machine. Interpretive and authorial affordances are coupled – a concern with the machine enables audience experiences that aren't achievable otherwise.

### 3.3 Combining Interpretive and Architectural Concerns

The splitting of AI-based art practice into interpretive and authorial concerns is for heuristic purposes only, as a way to understand how Expressive AI adopts concerns from both art practice and AI research practice. Expressive AI practice combines these two concerns into a dialectically related whole; the concerns mutually inform each other. The "interface" is not separated from the "architecture". In a process of total design, a tight relationship is maintained between the sensory experience of the audience and the architecture of the system. The architecture is crafted in such a way as to enable just those authorial affordances that allow the artist to manipulate the interpretive affordances dictated by the concept of the piece. At the same time, the architectural explorations suggest new ways to manipulate the interpretive affordances, thus suggesting new conceptual opportunities. Thus both the artist's engagement with the inner workings of the architecture and the audience's experience with the finished artwork are central, interrelated concerns for Expressive AI.

The AI-based artist should avoid architectural elaborations that are not visible to the audience. However, this admonition should not be read too narrowly. The architecture itself may be part of the concept of the piece, part of the larger interpretive context of people theorizing about the piece. For example, one can imagine building a machine like *Terminal Time* in which some small

collection of historical narratives have been prewritten. The narrative played is determined by a hard-coded selection mechanism keyed off the audience polls. For any one audience, the sensory experience of this piece would be indistinguishable from *Terminal Time*. However, at a conceptual level, this piece would be much weaker than *Terminal Time*. A *Terminal Time* audience is manipulating a *procedural process* that is a caricature of ideological bias and of institutionalized documentary filmmaking. The operationalization of ideology is critical to the concept of the piece, both for audiences and for artists and critics who wish to theorize the piece.

#### 4. The Code Machine and the Rhetorical Machine

AI (and its sister discipline Artificial Life), consists of both technical strategies for the design and implementation of computational systems, and a pared, inseparable, tightly entangled collection of rhetorical and narrative strategies for talking about and thus understanding these computational systems as intelligent, and/or alive.

These rhetorical strategies enable researchers to use language such as “goal”, “plan”, “decision”, “knowledge”, to simultaneously refer to specific computational entities (pieces of program text, data items, algorithms) and make use of the systems of meaning these words have when applied to human beings. This double use of language embeds technological systems in broader systems of meaning.



Figure 2. Total system = code machine + rhetorical machine

There is an uncomfortable relationship between a purely relational (and thus literally meaningless) technical manipulation of computational material, and the interpretation of this computational material by a human observer. Simon and Newell posited the physical symbol system hypothesis as a fundamental assumption of AI [16]. This hypothesis states that a physical system consisting of a material base that can take on various configurations (call these configurations “symbols”) and a material process that manipulates these physical constellations to yield new constellations is sufficient for the production of intelligent behavior. This formulation immediately produces an interpretation problem in which an external observer is necessary in order to view the material constellations as signs in such a manner that intelligence can be observed in the material production of sign from sign. Interpretation, with all of its productive open-endedness, is thus crucial to the definition of intelligent system, but is usually pushed to the background of AI practice.

The necessity of rhetorical strategies of interpretation is not avoided by “subsymbolic” techniques such as neural networks or genetic algorithms utilizing numeric genomes (i.e. not the tree-

shaped, symbolic genomes of genetic programming), nor by machine learning methods based on generalization from training data, nor by behaviorist robotic techniques that link sensors to effectors through stateless combinational circuitry or finite state machines. These approaches still require the interpretation of an observer in order to make sense of the input/output relationships exhibited by the system, to select the primitive categories (features) with which the inputs are structured, and to tell stories about the processes producing the input/output relationships. These stories are essential for thinking through which technical constructions to try next, that is, for simultaneously defining a notion of progress and a collection of incremental technical constructions that make progress according to this notion.

The rhetorical strategies used to narrate the operation of an AI system varies depending on the technical approach, precisely because these interpretative strategies are inextricably part of the approach. Every system is doubled, consisting of both a computational and rhetorical machine (see figure 2). Doubled machines can be understood as the interaction of (at least) two sign systems, the sign system of the code, and a sign system used to interpret and talk about the code.

The central problem of AI is often cast as the “knowledge representation” problem. This is precisely the problem of defining structures and processes that are *simultaneously* amenable to the uninterpreted manipulations of computational systems *and* to serving as signs for human subjects. This quest has driven AI to be the most promiscuous field of computer science, engaging in unexpected and ingenious couplings with numerous fields including psychology, anthropology, linguistics, physics, biology (both molecular and macro), ethnography, ethology, mathematics, logic, etc. This rich history of simultaneous computational and interpretive practice serves as a conceptual resource for the AI-based artist.

The relationship between the sign system of the code (the code machine) and the sign system used to talk about the code (the rhetorical machine) can be explicated via a semiological analysis. By semiology, I mean the semiotic tradition following Saussure’s General Linguistics [19], and explicated by thinkers such as [9, 4]. The treatment in this paper most closely follows Barthes [3, 4].

#### 4.1 The Code System

The program code, considered as a sign system, relates two planes: a plane of expression containing the space of all possible pieces of program text (the marks on a screen or page), and a plane of content containing the space of all potential executions. That is, a piece of program code is a signifier signifying (the mental concept of) the effect of executing this code. For example, the signified of the simple sign (code fragment)  $x = 1$  is, for programmers used to working in imperative languages, probably something like placing a 1 within a box labeled x.

Note that code signs, as is the case with any sign, provide no privileged access to an unmediated reality. The signified is the mental concept of an execution, not the execution itself. The relationship between the mental concept of an execution and the physical effect of executing a piece of code on a concrete computer (e.g. for contemporary digital computers, changing voltage levels in pieces of silicon) falls outside of the purview of structuralist semiotics. A code fragment is a sign-function, having both a utilitarian, technical use (the physical effect of executing

the code on a concrete machine), while serving as a sign for its potential execution. Obviously there are constraints imposed on sign value by use value; for example, the physicality of a rubber ball, and the technical functions (e.g. bouncing) that the physicality of a rubber ball supports, prevents (or at least makes quite difficult) the rubber ball from taking on the sign value of a tasty snack. Similarly, the possible sign values of a code fragment are constrained by the use value, the physical effect of its execution on concrete machinery. Though a structuralist semiotic analysis has its limits, such as difficulty in offering a detailed analysis of the relationships between sign and use value, it remains the case that much of human activity is structured by language-like interactions, from which a semiotic analysis gains its traction. In the specific case of the activity of programming, programmers think about potential executions and read and write texts to express those potential executions; this language-like activity suggests that the semiotic view of program code as a sign system, while not explaining *everything* about the human activity of programming, is likely to yield dividends.

To further unpack the idea of code as a semiotic system, consider the example of rhetorical goals in *Terminal Time*. The textual representation, the code, for a specific rhetorical goal appears in Figure 3.

```
(def-rhetgoal
  :name :give-positive-example-of-big-science
  :app-test
  (%and
    ($isa ?event %SciTechInnovationEvent)
    ($performedBy ?event ?bigsci)
    ($isa ?bigsci $LegalGovernmentOrganization)
    ($isa ?bigsci $ResearchOrganization))
  :rhet-plans (:describe-event)
  :emotional-tone :happy)
```

**Figure 3. The code representation of a rhetorical goal.**

This complex sign is itself a syntagm, composed of a constellation of signs. But considering the complex sign as a unity, the rhetorical goal signifies potential executions in which the system will tend to include a certain class of historical events in the constructed documentary, in this case, events in which governmental research organizations engage in scientific or technical research, in such a way as to make a certain point, in this case, that it is beneficial when science and government come together. It is interesting, perhaps surprising, that this relatively small textual signifier signifies potential executions that relate so directly to *Terminal Time's* output; watching a generated documentary (in which this goal is active) with this code sign in hand, it is possible to relate the appearance of specific historical events in the documentary (such as a breathless, glowing description of the moon landing or the invention of the atomic bomb) to this code sign, that is, to the effect on execution of this textual signifier. It is certainly not a given that a system of code signs would necessarily provide form to the plane of textual representations (expression) and the plane of potential executions (content) in this way. It takes work to articulate the planes in this particular way – this work is in fact the *creation of a custom code system*.

Standard languages, such as C++, lisp, or Java, define code systems, specific ways of chopping up the spaces of textual representations and potential executions. Like many sign-function systems, the more radical innovation of the creation of the sign

system lies with special individuals or organizations who define the language, with consumers of the language limited to working with the signs, the associations between text and execution, established by the language. But it is standard practice in computer science, enabled by Turing equivalence, to use a pre-given code system (language) to implement new code systems that provide *different associations between text and execution*. This practice allows individuals to engage in the more radical innovation of creating new code systems particularly suited for a specific task. Mainstream languages, such as the three mentioned above, tend to be strongly procedural; the control structure, which determines the temporal relationship between bits of execution, is explicitly captured in the textual representation. However, this is not the only kind of code system. One can define purely declarative code systems, such as the rhetorical goal above. In declarative systems, the textual representation does not explicitly capture temporal relations in execution. Rather, the code signs indicate execution propensities. The system as a whole will tend to behave in certain ways if the declarative sign is part of the system, though the precise execution path (temporal sequence of sign execution) is unknown. Or the custom language may be a hybrid, such as ABL, which combines the declarative features of production systems with the procedural features of more mainstream languages.

The *architecture* is the conglomeration of code that implements a custom language, that is, establishes the relationship between bits of textual representation and potential executions. For example, in *Terminal Time* a rhetorical goal becomes a sign by virtue of its role within the entire architecture. The rhetorical goal has relationships with or participates in many parts of the architecture, including the knowledge base, the story board (where narrative construction takes place), natural language generation, the selection of music, and (indirectly, through the goal's effect on the natural language generator) the sequencing of video clips. This little bit of text gains its meaning through its effect on a broad array of processes throughout the architecture.

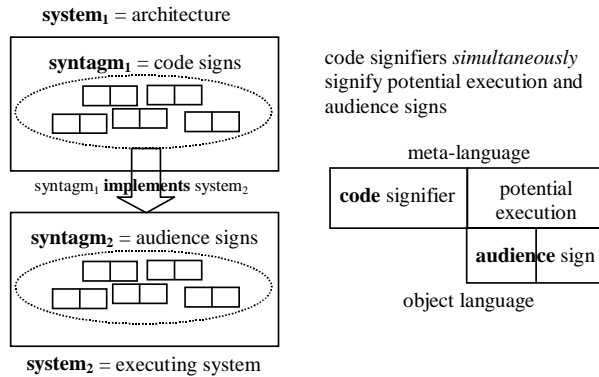
At this point it is possible to provide a semiotic account of the code system properties that yield interpretive and authorial affordances.

#### 4.1.1 Affordance in the Code System

An AI-based artwork is a semiotic system productive of a (potentially large) number of syntagms. AI-based artworks are thus *generative*; computational processes provide the combinatoric machinery necessary to select terms out of the fields of potential terms (associative fields) provided by the system. The system produces variable syntagms in different situations. For example, *Office Plant #1's* behavior over time depends on the email received by its owner, the content of documentaries generated by *Terminal Time* depends on audience answers to the psycho-graphic polling questions, and Trip and Grace's moment-by-moment behavior in *Façade*, as well as the more global story structure, depend on the player's real-time interaction and patterns of interaction over time.

The internal structure of the machine, the program code, wires, circuits and motors out of which a work might be constructed, is itself a syntagm of the semiotic system defined by the architecture (see Figure 4). The architecture consists of the custom code systems, processes, modules, and relationships between modules, which together define the implementation language, the sign

system within which the work will be constructed. Building an AI-based artwork thus means constructing a semiotic system of implementation (an architecture, system<sub>1</sub>) such that it supports the construction of a syntagm (the specific work built within the architecture, syntagm<sub>1</sub>), which, when executed, becomes a semiotic system (system<sub>2</sub>) autonomously productive of its own syntagms (syntagm<sub>2</sub>) in different situations. System<sub>1</sub> (the architecture) has appropriate *authorial affordances* when there is a “natural” relationship between changes to the syntagm<sub>1</sub> and changes in the syntagmatic productivity of system<sub>2</sub>. By “natural” is meant that it is easy to explore the space of syntagmatic productivity consistent with the artistic intention of the piece.



**Figure 4. Relationships in the code system.**

For example, in *Terminal Time*, the AI architecture is system<sub>1</sub>. Syntagm<sub>1</sub> is the collection of historical events (collections of higher-order predicate calculus statements), rhetorical goals, rhetorical devices, natural language generation rules, rhetorical plans, and annotated video and audio clips, which collectively make up the specific artwork that is *Terminal Time*<sup>1</sup>. Individual signs within syntagm<sub>1</sub>, as well as syntagm<sub>1</sub> as a whole, *are* signs (have meaning) by virtue of their participation within system<sub>1</sub>. The execution of syntagm<sub>1</sub> results in system<sub>2</sub>, in a runtime instance of *Terminal Time*. And, as the audience interacts with system<sub>2</sub>, it produces syntagm<sub>2</sub>, a particular documentary out of the space of all possible documentaries expressible within (producible by) system<sub>2</sub>. While the structure of syntagm<sub>2</sub> is quite literally determined by system<sub>2</sub>, for the audience, the meanings expressed by syntagm<sub>2</sub> are determined by a meshwork of different sign systems, including the system of documentary imagery, the system of cinematic music, the linguistic system for English (the voiceover), and a folk psychology of the execution of system<sub>2</sub> (e.g. “we voted that religion is a problem in the world, and now it’s trying to make the point that religion is bad”). Thus syntagm<sub>2</sub> is multi-articulated; its meaning is determined not just by system<sub>2</sub>, but also by a number of sign systems *outside* the technical system<sub>2</sub>.

System<sub>1</sub> is a meta-language for talking about system<sub>2</sub>; utterances in system<sub>1</sub> (syntagm<sub>1</sub> or fragments) talk about potential utterances of system<sub>2</sub> (syntagm<sub>2</sub> or fragments) (see Figure 4). For *Terminal Time*, system<sub>1</sub> utterances, such as the rhetorical goal in Figure 3,

are a way of talking about potential system<sub>2</sub> utterances, such as a breathless, glowing description of the invention of the atomic bomb. System<sub>1</sub> offers *effective authorial affordances* when one and the same syntagm<sub>1</sub> simultaneously talks about desired syntagms<sub>2</sub> (or fragments), and, when executed, implements the appropriate system<sub>2</sub> that indeed produces the desired syntagms<sub>2</sub>. This property is not trivial – there are a number of ways in which it can fail to hold.

It can be the case that system<sub>1</sub> fails to provide appropriate signs for talking about desired properties of syntagm<sub>2</sub>. For example, an early version of *Terminal Time*’s architecture represented historical events directly at the natural language generation and video clip sequencing level. There was a fairly direct connection between answers to the audience polls and the generation of specific text about specific events. Given this system<sub>1</sub>, it was impossible to express *general* relationships between poll answers and categories of events. For example, if the winning answer to the question “What is the biggest problem in the world today” is “It’s getting harder to earn a living and support a family”, the desired syntagm<sub>2</sub> should include events demonstrating the evils of capitalism. Given a relatively direct connection between poll answers and natural language generation, there just was no way of expressing this more general desired property of syntagm<sub>2</sub>, and thus certainly no way of implementing the appropriate system<sub>2</sub> with syntagm<sub>1</sub>.

It can be the case that syntagm<sub>1</sub> utterances *purport* to talk about desired syntagms<sub>2</sub>, but in fact, when executed, don’t implement a system<sub>2</sub> that produces the desired syntagm<sub>2</sub>. For example, in *Office Plant #1*, statistical text classifiers map incoming email into social and emotional categories. The categories appearing in an email stream then condition the physical behavior of the device. However, if the email categories are being inappropriately assigned to individual emails, then the decision making process that uses the assigned categories to decide which physical behaviors to perform will make inappropriate decisions. That is, the author will think that they’re specifying a system<sub>2</sub> that reacts in a specific way to, for example, an *apology* email, when in fact the internal label *apology* (a sign in syntagm<sub>1</sub>) does not properly correspond with the intuitive notion of an apology. Thus the statistical text classifiers must be trained in such a way that the labels (categories) produced by the classifiers have an appropriate correspondence with email messages.

As a final example of the failure of authorial affordance, it can be that case that syntagm<sub>1</sub> is successful in simultaneously describing a desired syntagm<sub>2</sub> and implementing an appropriate system<sub>2</sub>, but that, when the audience (who may in fact be the same as the author) actually experiences the produced syntagm<sub>2</sub>, its interpretation is different than expected. This situation arises precisely because syntagm<sub>2</sub> doesn’t participate in just the technical system<sub>2</sub>, but in a meshwork of sign systems *outside* of the technical system. That is, part (perhaps a large part) of the meaning of syntagm<sub>2</sub> is opaque to the technical system, but rather comes along for the ride as the technical system manipulates and produces signs. For example, in *Façade*, a beat, and the associated beat behaviors, may purport to serve the dramatic function of communicating that when Trip asked Grace to marry him she wasn’t really ready, while simultaneously communicating that they are both getting more upset and that Grace currently feels disaffiliated with the player. The associated beat code may simultaneously describe the author’s vision of the desired run-

<sup>1</sup> Since signs may be added or changed over time, such as the modification or addition of rhetorical devices or historical events, *Terminal Time* as a specific piece changes over time.

time experience, and, when executed, implement the author's vision of the desired runtime experience. But when the author, or another player, plays the experience, Trip and Grace actually seem less upset than in the preceding beat, even though they are supposed to be more upset. What happened here is that the details of the writing, and how the details of their physical performance actually read, are *extra-architectural*; they lie outside the literal code of the system. Even though the beat is "performing to spec", other sign systems are subverting its interpretation. Every AI system is doubled. A description of the code system is not enough – we need to examine the rhetorical system.

## 4.2 The Rhetorical System

The signs of both system<sub>1</sub> and system<sub>2</sub> are multi-articulated; their meaning arises both from syntagmatic and paradigmatic constraints established by the respective code systems, but also from a collection of sign systems *outside* of the code systems. This collection of external code systems is the rhetorical system. Both authors and audiences make use of the rhetorical system in narrating the operation of the system and forming intentions with respect to the system. The code and rhetorical systems are tightly entangled; both play a role in understanding interpretive and authorial affordances.

### 4.2.1 (Audience) Interpretive Surplus

Syntagm<sub>1</sub> never completely describes all the properties of syntagm<sub>2</sub>; though system<sub>2</sub> literally prescribes the possible elements (paradigm) and spatial and temporal relationships between elements (syntagm) of syntagm<sub>2</sub>, a portion (perhaps a large portion) of the signification is determined by external sign systems. This interpretive surplus occurs because system<sub>2</sub> operationalizes a meta-language (syntagm<sub>1</sub>) for describing the audience experience (syntagm<sub>2</sub>). The signifieds of this meta-language are themselves signs, participating in external sign systems, which are handled by the meta-language.

The crafting of these external, handled signs, becomes an irreducible problem in design and aesthetics. These handled signs must be crafted to marshal the signifying resources of these external sign systems in such a way as to match the purported meanings of the code system. For example, in *Façade*, we as authors have to write dialog that consistently communicates the character of Grace and Trip, while communicating meanings appropriate for a specific beat goal within a specific beat, while also being re-sequencable to various degrees. Specific lines of dialog must meet multiple constraints established by how the code machine will make use of the line. Additional meaning is carried by how a voice actor performs the line. The nuances of emotional tone, irony, sarcasm, desperation, etc., communicated by the voice performance, must also be consistent with these constraints. In authoring *Façade*, there is a reciprocal process between authoring these handled signs (e.g. dialog, snippets of animation data) and code-level authoring within the architecture. Consistency between handled signs and manipulation by the code machine is established by moving back and forth in the authoring of these two domains. But consistency is not the same as identity; there are always aspects of audience interpretation that escape the code machine.

Another avenue for interpretive surplus is connotation; the handled signs may become the plane of denotation for a connotative system. For example, in *Terminal Time*, the

ideological arguments made by the system are often (purposely) undermined through irony. The details of imagery, music, and the narrative track connote irony, while at the level of denotation an earnest argument is being made. For example, if the anti-religious rationalist ideologue has been activated, a 20<sup>th</sup> century event it may make use of is the Chinese invasion of Tibet. Within the knowledge base, the two actors of this event are Tibetan Buddhists (which the system infers are a kind of Religious Group), and Maoists (which the system infers are a kind of Rationalist through their connection to Marxism). Furthermore, the event is a War, instigated by the Maoists (Rationalists) against the Buddhists (Religious Group), in which the Maoists are successful. This is enough for the Anti-Religious Rationalist to decide it can use this event as a Positive Example of Rationalist Progress. Assuming that this event spin (the ideologically-slanted representation of the "objective" representation in the knowledge base) makes it into the final generated documentary, the system will earnestly argue that this is a positive example of Rationalists mopping up the remaining dregs of irrational religion (e.g. "There were reports that Buddhists monks and nuns were tortured, maimed and executed. Unfortunately such actions can be necessary when battling the forces of religious intolerance.") over a montage of Tibetan Buddhist imagery and Chinese soldiers holding monks at gunpoint, while playing the happy, "optimistic" music loop. The system does not "know" that it is undermining its argument through irony; irony is not a property described within the code machine. We as system authors marshaled the handled signs (language, video clips, music) to connote irony on top of the structure explicitly provided by the code machine.

Given that the audience interpretation of syntagm<sub>2</sub> always escapes full specification by the code machine, it may be tempting to conclude that computer-based art practice should primarily make use of the signifying resources of external sign systems via handled signs. Crafting the handled signs, animation snippets, imagery, video clips, music loops, and so forth, falls comfortably in the realm of more traditional art practice. Such an approach would move back towards the "code as a hack" model, throwing together the minimum code machine necessary to coarsely manipulate handled signs. But this approach would severely compromise the intentional affordances. As the interpretive surplus becomes larger and larger, with more of the interpretive affordance pushed onto the handled signs, an imbalance grows between the intentional affordances offered by the system and the system's ability to actually respond to these intentions. The rich handled signs suggest many avenues of action to the audience. But with no corresponding richness in the code machine, there is no way for the work to respond to these actions; the rich, coarsely handled signs suggest a richness of response that the work can't satisfy. But the reason for designing a rich and expressive architecture goes beyond the "utilitarian" goal of supporting audience agency. The architecture (system<sub>1</sub>), and systems designed within it (syntagm<sub>1</sub>), are themselves embedded in a meshwork of external sign systems, providing the AI-based artist with a rich architectural surplus.

### 4.2.2 Architectural Surplus

Agre [2] describes how AI technical practice provides narrative affordances that support AI researchers in creating stories describing the system's operation.

... the practical reality with which AI people struggle in their work is not just “the world”, considered as something objective and external to the research. It is much more complicated than this, a hybrid of physical reality and discursive construction. ... Technical tradition consists largely of intuitions, slogans, and lore about these hybrids, which AI people call “techniques”, “methods”, and “approaches”; and technical progress consists largely in the growth and transformation of this body of esoteric tradition. [2:p. 15]

Different practices (e.g. classical AI, interactionist AI) provide different affordances for narrating system behavior. For the classical AI researcher, the discursive construction consists of ways of talking about “goals”, “plans”, and “knowledge”, while for the interactionist AI researcher, the discursive construction consists of ways of talking about “embodiment”, “action”, and “improvisation”. These discursive constructions are a necessary part of the functioning of the system.

To understand what is implied in a claim that a given computer model “works”, one must distinguish between two senses of “working”. The first, narrow sense, again is “conforms to spec” – that is, it works if its behavior conforms to a pre-given formal-mathematical specification. ... the second, broad sense of “working” ... depends on specific words of natural language. As I mentioned at the very beginning, an AI system is only truly regarded as “working” when its operation can be narrated in intentional vocabulary, using words whose meanings go beyond mathematical structures. When an AI system “works” in this broader sense, it is clearly a discursive construction, not just a mathematical fact, and the discursive construction succeeds only if the community assents. [2:p. 14]

In typical AI research practice, these affordances are often not consciously acknowledged or manipulated. Rather, they serve as part of the unconscious background, co-evolving with the technical practice as a silent but necessary partner in the research. Systems are spoken of as having “goals” or engaging in “embodied action”, as if these were primitive, readily detectable properties, like being blue, or being cold, rather than the hard-won results of rhetorical construction and debate. But in Expressive AI practice, these discursive constructions are an explicitly manipulated resource, an architectural surplus that makes the architecture not just a bunch of code, but a way of thinking about the world.

Within the semiotic framework of this chapter, the architectural surplus (an interpretive surplus on the author side), can be understood as one or more meta-languages, in which the signs in system<sub>1</sub> (syntagm<sub>1</sub>) form the content plane, and as one or more connotative systems, in which signs in the meta-language form the plane of denotation.

For example, consider joint goals in ABL. The code sign for a joint goal appears in Figure 5. The sign signifies that a team of ABL agents will attempt to achieve Goal1(). A meta-language allows us to talk about and thus operate on these code signs. This meta-language consists of ordinary language that has been co-opted into talking about code signs. This meta-language in turn serves as the plane of denotation for a connotative sign system – this connotative sign system contains the “spillover” of the co-opted ordinary language, connotative meanings that escape the

strict meaning of the code signs. In this case, the meta-language sign for a joint goal connotes the idea of a team of people working together, with all the non-formalized richness of this notion. The connotation lifts the code sign out of the circumscribed meaning provided by the architecture, and into the more open-ended sign system used to talk about coordinated human activity in the everyday world. Once lifted into this connotative system, the author can use the connotative sign system to think about the human realm of teamwork. But new signs reached by thinking in the connotative plane can in turn have signifiers in the meta-language whose signifieds lie back in the code system. Thus ordinary language, in this case the ordinary language of human teamwork, becomes a meta-language for talking about and manipulating a technical system, in this case the code system for joint goals in ABL. This movement, from code system, into ordinary language, and back into code system, creates a circulation of signs that suggests both new ways of using the architecture and new architectural elaborations, in this case new ways of using joint goals and new architectural elaborations for joint goals.

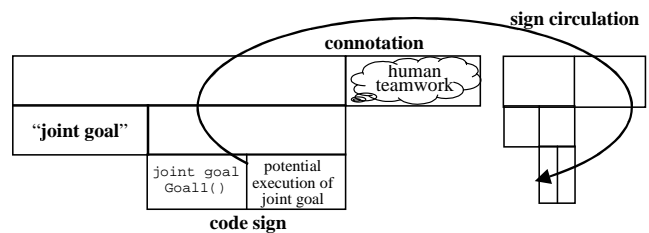


Figure 5. Code signs, meta-language, and connotation.

Consider first how the ordinary language system of human teamwork suggests new ways of using joint goals. In the everyday human world, we think of people coordinating to achieve goals they want to achieve; that is, we imagine people having a positive emotional valence towards a goal. Two people might team up to hang a picture, or change a tire, but we don’t picture people teaming up to have a big fight, or teaming up to accomplish a mugging, with one team member the victim and one team member the mugger. An author may thus never think of using joint goals to coordinate a big fight among two agents. But now imagine that in the connotative plane we start thinking about teams of movie actors or stage actors. In acting within a play or movie, human actors often tightly coordinate in the carrying out of dramatic activity in which the characters strongly oppose each other, as in, for example, a play in which a marriage falls apart as years of buried frustrations and misunderstandings are revealed. Now this gives us the leverage (meta-language) to imagine using joint goals in *Façade* to tightly coordinate conflicts between characters. Ordinary language, used as both the plane of connotation and as meta-language, is a necessary part of the *total* system – it provides the code system with broader meaning and consequently suggests new ways of manipulating the code system. Note that this example involves consciously manipulating and exploring the plane of connotation in order to reveal a new possibility within the code system. If we were uncritically wedded to the ordinary language system of “rationality”, in which people only pursue goals for things they emotionally desire, then the code system idea of jointly accomplishing conflict may never arise.

The plane of connotation and meta-language not only suggests ways of using the code system (syntagm<sub>1</sub>), but modifications and elaborations of the architecture itself (system<sub>1</sub>). Continuing with the joint goal example, consider the control of activity within a joint goal. In ordinary language, when we imagine team members accomplishing a task together, we often imagine the decision of what step to do next being distributed among the team members. Certainly there are hierarchical situations in which a team leader is responsible for managing the teams, but many teamwork situations are more collaborative and decentralized. Now consider the initiation of joint goals in the code system. When one member of a team initiates the joint goal, the other members of the team, on successful entry into the joint goal, spawn the goal at the root of their active behavior tree (ABT). Only the joint goal initiator has the goal deeper within the ABT. If other members of the team initiate joint subgoals in the service of the original joint goal, these joint subgoals will appear at the original initiator's ABT root. This is a bit counter-intuitive, given that within the ABT subgoals are normally children of the goal (via a behavior) they are in service to. But strictly at the code level there is nothing wrong with this arrangement. However, consider how the ABT is connotatively read or interpreted. The ABT captures the structure of an agent's thoughts, its mind. It is not just a bookkeeping mechanism controlling execution, but a *representation* of the agent's activity. Reflective processes (meta-behaviors) may treat the ABT directly as a representation. But even without reflection, the mechanisms for success and failure propagation, the many annotations that modify success and failure propagation, and continuously monitored conditions, all work together to support the reading of the ABT as a representation. When a goal appears deep in the ABT, it is enmeshed in more complex patterns of activity than a goal shallower in the ABT – ABT depth becomes a proxy measure for the complexity of the agent. With this reading of the ABT, combined with the ordinary language model of teamwork, the default joint goal initiation mechanism is seen as lacking. Initiated joint goals, since they are always at the root of the ABT, aren't able to fully participate in complex patterns of activity. This is particularly problematic for "flat" teams, in which all team members equally participate in the control logic for the team, and thus both initiate and respond to requests to enter joint goals. This circulation between readings of the ABT, code signs for joint goals, and readings of these code signs, suggests an architectural modification supporting the initiation of joint goals anywhere in the ABT.

Authorial affordance consists not just of the code system relationship that syntagm<sub>1</sub> simultaneously implements system<sub>2</sub> and describes syntagm<sub>2</sub>, but also of the rhetorical relationship that syntagm<sub>1</sub> is readable and handleable by interpretive systems and meta-languages. An architecture is a machine to think with. The complex circulation between code signs and the interpretive framework provides authors with both resistance (some things will appear hard or impossible) and opportunity (new ideas arise). Thinking with the architecture suggests new audience experiences, creating a feedback loop between authorial intention and the details of the *total* system (code + rhetoric). But establishing this interpretive framework, the plane of connotation and meta-language, takes real work. It is the outcome of a practice that simultaneously tries to articulate the code machine *and* the ways of reading it and talking about it. In contrast, a practice that views the system as a hack, as a means to an end, will likely

construct systems with poor authorial affordances, lacking both the code system relationships and rich rhetorical frameworks necessary to enable new audience experiences.

### 4.3 Idioms

Idioms are ways of using an architecture, conventional structures for the authoring of syntagm<sub>1</sub>. Idioms arise through the interplay of the architecture and its interpretive frameworks. In a sense, the idioms actually cash out the interpretive framework, being the place where interpretation and code meet. This is why idioms are so important for truly understanding an architectural system. An abstract description of a code system will make use of all kinds of ordinary language words, such as "plan", or "embodied activity", or "learning", but understanding the particular entanglement of rhetoric and code that is the total system requires examining the detailed circulation between these language signs and code signs. Idioms are the place where this detailed circulation occurs.

As idioms become larger and more diffuse, they begin restricting the circulation between code and rhetoric. The code signs become large and diffuse, making the connotative lifting and meta-language handling difficult. Idioms can thus reveal breakdowns in the total system, conceptual domains in which the circulation between rhetoric and code are restricted. The breakdowns suggest architectural opportunities, modifications of the architecture that enable new idioms and simultaneously re-articulate the interpretive sign systems, providing new ways of talking and thinking about the code system. Systems built without an explicit concern for authorial affordances are likely to be *all* idiom, and thus severely restrict the circulation between rhetoric and code. This would be the case, for example, if *Facade* was written as a giant program in a standard programming language such as C. The only code signs at our disposal would be the rather low-level signs provided by C. Everything else would be idiom, with large chunks of C code having only a diffuse relationship to signs of the audience experience (syntagm<sub>2</sub>) and to connotative and meta-languages. This extreme case of the code system being nothing but idiom, code piled on code, provides poor authorial affordances, making it difficult to think about, discover, and express, new conceptual frameworks and new audience experiences.

### 4.4 Generality of the Doubled Machine

The use of a structural semiotic terminology in this chapter, with the focus on "sign systems", "languages", "connotation" and so forth, may lead a reader to conclude that the analysis of affordances in terms of doubled machines of rhetoric and code is only useful for classical AI systems, with their explicit focus on symbolic knowledge. The analysis applies much more broadly than this, however, to any AI or ALife practice. All such practices make use of a rich entanglement between technical systems and ways of talking and thinking about the technical system. Consider a robot built along the lines of subsumption architecture [6], in which finite state machines mediate rather directly between sensory input and motor actuation. The finite state machines may in fact be implemented entirely in hardware, rather than as code in a general purpose micro-controller. Yet there is still a "code machine" that participates in complex discursive constructions. Wires bearing voltages are in no less need of interpretation than fragments of textual code, and participate in the same sign system relationships that support interpretive and authorial affordances.

The focus in this chapter on authorship may similarly lead a reader to conclude that this analysis is not relevant to machine learning. But again, the methods of machine learning consist of a technical/rhetorical system, one organized around the “learning” or “discovering” of “patterns” in “raw data”. But, of course, human authors select the primitive features, define the representations of hypotheses or distributions, define the search methods employed to tune parameters, and design how particular machine learning methods are embedded in larger architectures. For example, *Office Plant #1* makes use of the technical/rhetorical system of text learning as part of an architecture supporting the creation of a non-human companion responding to email activity.

## 5. Conclusion

This paper develops authorial and interpretive affordances as central terms in the hybrid practice of Expressive AI. The relationship between these two affordances shows how Expressive AI is simultaneously concerned with art’s creation of meaningful experience (and the consequent focus on interpretation of the art object), and AI’s construction of machines that can be understood as behaving intelligently (and the consequent focus on the structures, properties and processes of these machines). Structuralist semiotics, through its concern with sign systems and the relationships between systems, provides a common ground in which both the artwork as experienced by the audience and the construction of machines as experienced by the author can be seen as instances of sign systems – this provides the framework for a more detailed analysis of the relationship between these affordances.

As an analytical framework, structuralist semiotics has its limits. Arising from the tradition of Saussure, its view of the world as a meshwork of language systems whose rules can be analyzed has trouble accounting for the actual processes involved in the use and production of signs. Some work in the analysis of computational media has fruitfully made use of Peircean semiotics, whose sign concept includes a notion of meaning more amenable to process (e.g. [1, 7:chapter 4]). Further analysis of the negotiation of meaning in technical systems could fruitfully make use of ethnographic and phenomenological frameworks. However, the structuralist analysis here, with its focus on the relationships between sign systems, goes a long way towards understanding both how and why Expressive AI is simultaneously concerned with the code system and audience interpretation.

## 6. REFERENCES

- [1] Anderson, P. B., Holmqvist, B., Jensen, J. F. 1993. *The Computer as Medium*. Cambridge: The Cambridge University Press.
- [2] Agre, P. 1997. *Computation and Human Experience*. Cambridge, UK: Cambridge University Press.
- [3] Barthes, R. 1972. *Mythologies*. (Trans: Annette Lavers). New York: Hill & Wang. Translation of *Mythologies*, first published in 1957.
- [4] Barthes, R. 1967. *Elements of Semiology*. (Trans: Annette Lavers & Colin Smith). New York: Hill & Wang. Translation of *Eléments de Sémiologie*, first published 1964.
- [5] Boehlen, M., and Mateas, M. 1998. *Office Plant #1: Intimate space and contemplative entertainment*. Leonardo, Volume 31 Number 5: 345-348.
- [6] Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1).
- [7] Frasca, G. 2001. *Videogames of the Oppressed: Videogames as a Means for Critical Thinking and Debate*. Masters Thesis, Interactive Design and Technology Program, Georgia Institute of Technology. Available at: [www.ludology.org](http://www.ludology.org).
- [8] Gibson, J. 1979. *The ecological approach to human perception*. Boston: Houghton Mifflin.
- [9] Hjelmslev, L. 1961. *Prolegomena to a Theory of Language* (trans. Francis J Whitfield). Madison: University of Wisconsin Press. Translation of *Omkring Sprogteoriens Grundlæggelse*, first published in 1943.
- [10] Mateas, M. 2002. *Interactive Drama, Art and Artificial Intelligence*. Ph.D. Thesis, Computer Science Department, Carnegie Mellon University. CMU-CS-02-206.
- [11] Mateas, M. 2001. Expressive AI. *Leonardo: Journal of the International Society for Arts, Sciences, and Technology*, 34 (2), 147-153.
- [12] Mateas, M. and Stern, A. 2003. *Integrating Plot, Character and Natural Language Processing in the Interactive Drama Façade*. *Proceedings of Technology for Interactive Digital Storytelling and Entertainment (TIDSE) 2003*.
- [13] Mateas, M. and Stern, A. 2000. *Towards Integrating Plot and Character for Interactive Drama*. In *Working notes of the Social Intelligent Agents: The Human in the Loop Symposium*. AAAI Fall Symposium Series. Menlo Park, CA.: AAAI Press.
- [14] Mateas, M., Vanouse, P., and Domike S. 2000. *Generation of Ideologically-Biased Historical Documentaries*. In *Proceedings of AAAI 2000*. Austin, TX, pp. 236-242.
- [15] Newell, A. 1982. The Knowledge Level. *Artificial Intelligence* 18: 87-127.
- [16] Newell, A. and Simon, H. 1976. Computer science as empirical enquiry: symbols and search. *Communications of the ACM*, 19:113--126.
- [17] Norman, D. 1988. *The Design of Everyday Things*. New York: Doubleday.
- [18] Penny, S. 2000. Agents as Artworks and Agent Design as Artistic Practice. In K. Dautenhahn (Ed.), *Human Cognition and Social Agent Technology*. Amsterdam: John Benjamins.
- [19] Saussure, F. 1974. *Course in General Linguistics*. London: Fontana. Translation of *Cours de linguistique generale*, first published in 1916.
- [20] Turing, A. 1950a. Computing machinery and intelligence. *Mind* 59:433-60.