In Proceedings of the Sixth International Conference on Cognitive Modeling, 249-254.
Mahwah, NJ: Lawrence Earlbaum.

# Letter Spirit: A Model of Visual Creativity

**John Rehling (rehling@andrew.cmu.edu)**
Carnegie Mellon University, 5000 Forbes Avenue
Pittsburgh, PA 15213 USA

**Douglas R. Hofstadter (dughof@cogsci.indiana.edu)**
Center for Research on Concepts and Cognition
Indiana University
510 North Fess Street
Bloomington, IN 47408 USA

**Abstract**

The Letter Spirit program is a model of human creativity in the domain of typeface design. Creativity is involved in most human activities, but it is by its nonquantifiable nature at odds with many modeling approaches. The Letter Spirit program models the inherently creative task of typeface design with a two-level approach. The top level loop coordinates three modules that create letterforms, evaluate them aesthetically, and decide which letterforms need to be re-designed, until all 26 lowercase letters have been rendered suitably. On the lower level, each module carries out its task by modeling the way that subcognitive pressures combine and conflict to produce emergent behavior on the cognitive level. While certain types of reasoning can be modeled with the manipulation of symbols whose meaning is considered to be absolutely fixed, this work helps to illustrate why it is difficult to demonstrate creative behavior with systems of that kind.

## Introduction

The Letter Spirit domain is based upon a grid consisting of 56 quanta, each of which is a horizontal, vertical or diagonal line segment connecting adjacent points in a 3x7 grid. Some sample *gridletters* are displayed later in Figure 1.

### The task

The task of Letter Spirit is to accept as input a few gridletters from a human designer, with those *seed* letters intended to represent the same style. Using the seeds as a beginning, Letter Spirit creates versions of the remaining lowercase letters of the roman alphabet until it has produced an entire *gridfont* of 26 stylistically consistent gridletters. Letter Spirit can also begin a new gridfont from scratch, beginning without any human-created seeds and those gridfonts are usually stylistically coherent and are sometimes stylistically interesting

## Implementation

Letter Spirit consists of three modules, each being a relatively complex program solving a vital subtask of gridfont design, plus the top-level loop that coordinates the three modules into a single strategy of design called *review-and-revision*. In order to carry this strategy out, the program has one module, the Drafter, that, given a particular letter category (e.g., 'b') and a particular style (usually exemplified for the program by means of several examples of different letter categories in that style) creates an attempt at a gridletter that is a good representative both of that letter category and that style.

Superficially, it may seem sufficient to create but one such attempt per letter category until the gridfont is complete. Creating one attempt according to a prescribed method of creation is the approach of many computational models of creativity, but when the output of such a method is of low quality (as if often the case), the program consequently shows the brittleness that is characteristic of many efforts in AI. To get around this problem, Letter Spirit hands the Drafter's output to the other modules, called the Examiner and the Adjudicator, and they rate each Drafter-created gridletter for how well it fits its intended letter category and the intended style, respectively.

### The FARG approach

#### Foundations

While the three modules differ significantly in their implementation (that is inevitable, given their very different subtasks), they are all based upon an architecture used in other cognitive models that have been implemented by the Fluid Analogies Research Group. As the name implies, those efforts were primarily concerned with modeling how people create analogies by means of concepts that are fluid, defined in terms of norms that may or may not be upheld rather than rigid, strictly-enforced rules. Most of those projects, including Copycat (Mitchell, 1993), Tabletop (French, 1992), and Metacat (Marshall, 1999) aimed at implementing increasingly refined models of analogy. One of the Letter Spirit modules, the Examiner (McGraw, 1995) pursues a similar approach, although its task is gridletter categorization rather than analogy.

The FARG programs all have some common architectural elements, loosely inspired by the Hearsay-II speech recognition architecture of Erman, et al. (1980).

#### The Coderack: Fine-grained processing

In the FARG architecture, the counterpart to what would be, in a traditional computer program, a procedure call is a

*codelet*, a relatively short routine that performs some small operation. No individual codelet does very much of the program's work. They are intended to correspond meaningfully to small-scale cognitive events. The Coderack is the repository of codelets, and execution begins with the nondeterministic selection of one codelet from the Coderack. The selection process is nondeterministic, weighted by the individual *urgency* value attached to each codelet when it is posted to the Coderack. The selected codelet is removed from the Coderack and then executed. This activity is repeated until either the Coderack is empty or some other condition for halting is reached. A run begins with the placing of some codelets on the Coderack. As codelets are run, the number of codelets remaining on the coderack generally decreases, but codelets may, as part of their work, add more codelets to the Coderack. The length and composition (in terms of what types of codelets are present) of the Coderack both change over the course of a run

With each individual codelet performing only a very small portion of the overall task, FARG models progress towards producing an output only through the cumulative action of many codelets. As a spectator, one may infer implicit high-level goals in the activity of the Coderack although such goals are not explicit anywhere in the Coderack, and instead emerge as trends in the activity carried out one codelet at a time on a lower level. Multiple high-level activities can chronologically overlap as their associated codelets run in an interleaved order. That leads to implicit parallelism and, consequently, behavior that allows multiple high-level pressures to compete or cooperate as the program progresses towards a final answer.

Systems whose behavior emerges from the combined action of many small agents have been inspired by and likened to the work of an anthill as carried out by many ants, or the activity in a cell, with countless proteins each performing a tiny part. The benefits can be understood by means of an analogy with integral calculus. One of the insights underlying calculus is that the area under a curve can be approximated by fitting rectangles onto the Cartesian plane, between the curve and the x-axis, and calculating the sum of their areas. The thinner the rectangles, the better the approximation to the actual area bounded by the curve. In a Coderack architecture, likewise, the small grain size of processing by codelets offers the potential for a tight fit to the mental processes that the architecture models.

**Long-term and short-term memory**

Each FARG model has a short-term memory called the *Workspace* that consists of structured representations that relate how the symbols contained therein relate to one another. Further details vary widely from model to model.

Many of the models also have a localist connectionist network that allows (among other operations) a scheme of spreading activation to mediate, throughout the course of a run, how important concepts (represented by nodes in the network) appear to be to processing at any given point in time. In the case of Copycat, Tabletop, and Metacat, the network is called a *Slipnet*, which is imbued with some sophisticated characteristics that are not implemented in any of the Letter Spirit modules, and will not be described more in this paper. Letter Spirit's Examiner module has a simpler *Conceptual Network* that is a conventional localist connectionist network (McClelland and Rumelhart, 1981) with nodes on one level for whole letters and nodes on another level for letter parts; during a run, activation can spread between nodes so that when the relevance of one concept has been detected or suspected, the program can suspect the presence of related concepts. For example, when 'i' is activated, activation spreads to the 'dot' node as well, and future processing is then more likely to see any appropriate structure that is present *as* a dot.

Long-term memory is also important for each model, with the repository of necessary domain knowledge included in a store called the *Conceptual Memory*. The details of this also vary greatly from model to model.

**Parallel terraced scan**

The activity of a FARG model can be seen as the investigation of multiple competing possible answers to the same problem or subproblem, frequently with more than one possible answer being considered at the same time. A goal of the FARG architecture is to create models that direct effort preferentially towards answers that are apparently more promising. The *parallel terraced scan* describes the way in which a program can explore many possible answers at once (thus, in parallel) but devotes more time to pursuing those answers that seem more worthwhile (thus, terraced).

The result of a well-implemented parallel terraced scan is a program that finds good answers without having to commit equal amounts of computational resources to all possible answers. In general, a Coderack program implementing the parallel terraced scan carries out search (if its activity is to be described in that way) very differently from either a traditional search algorithm or a generate-and-test algorithm that is applied exhaustively to the set of possible answers. In a FARG model, good answers (not necessarily perfect answers) bubble up to the top naturally. The effect of the parallel terraced scan is to makes models of cognition more efficient while also raising the quality of their output. Decisions regarding how much computational effort should be directed towards each answer are made implicitly by the program during runtime, sensitive to the conditions of the particular run, rather than being made by the programmer, during programming time, and insensitive to the conditions of the future run. The parallel terraced scan is concerned with making a program that is intelligent not only with regard to its eventual output, but also in regard to how it goes about producing that output.

The details of how a FARG model can incorporate the parallel terraced scan vary according to the other details of the model. The process of imbuing a model with the parallel terraced scan (or increasing the *extent* to which it utilizes the parallel terraced scan) is documented in our account of how

the Examiner was optimized from its original implementation (Rehling and Hofstadter, 1997).

## Representing letters

People do not conceptualize letters as exact graphical shapes. If they did, deviations from the definition would be seen as inherently flawed. In reality, there is no one "right" font, although some certainly seem more normal than others.

Work on the Letter Spirit project has been guided by the belief that letters are represented by roles – abstract concepts on a level beneath that of letters. The constituent strokes from which children learn to draw letters approximately indicate the level of roles. Letters are defined by one or more role-sets, which specify which roles make up the letter and how they ought to touch or otherwise relate spatially to one another. An actual physical rendering of a letter – a *letterform* – can be decomposed into physical *role-fillers*, which satisfy (to some degree) the definitions of their corresponding roles. It is important to note that role-fillers are exact shapes, but roles are not. Roles are concepts, flexible enough that a great variety of shapes may serve as role-fillers for them. The role hypothesis and support for it is explained in greater detail in (McGraw, 1995).

## Representing style

### Stylistic properties

The work on Letter Spirit aims to capture a reasonable range of styles by implementing (with routines allowing for perception as well as creation) a number of stylistic properties, which can define many kinds of gridfont, though not all of the ones that people can invent. This implementation of Letter Spirit realizes three kinds of stylistic property, and diverse styles are represented as combinations of these.

One type of stylistic property is *motifs*. A motif is a particular shape that recurs in numerous gridletters. "Shape" can be thought of in a number of levels of literalness. Most literal would be a set of contiguous quanta with precise location on the grid specified. Less literal versions allow translated, reflected, or rotated versions to count as the same shape. Letter Spirit allows for different levels of literality in motifs, with an emphasis on detecting and using those motifs that are inherently more noteworthy. The larger and more literal a motif is, the more noteworthy it for it to occur.

*Abstract rules* are properties that may be present in an individual gridletter. These are "Thou shalt not" rules that forbid quanta of certain orientation, angles of certain measure, quanta within certain zones of the grid, and collinear stretches of segments of various lengths.

The third kind of stylistic property is that which occurs when role-fillers deviate significantly from their corresponding roles; these conflicts are called *norm violations*. If a role-filler lacks a particular property expected for its abstract role, that does not destroy its membership in the category, although it makes it a less prototypical member of the category.

### The Thematic Focus

The style of the gridfont that Letter Spirit is in the process of designing is represented in a structure called the *Thematic Focus*. The Thematic Focus consists of a handful of levels (six) with different levels indicating the degree to which the SPs contained in that row are important to the style of the gridfont. Being located on a higher level indicates that a stylistic property is more important. When a stylistic property has been found in one gridletter, it is placed on the lowest level of the Thematic Focus. Promotion (and demotion) of stylistic properties to higher (and lower) levels throughout a run is probabilistic, so that a stylistic property's frequency of occurrence in the gridfont roughly correlates with how high in the Thematic Focus it is located. The higher in the Thematic Focus that a stylistic property is found, the greater its influence in the subsequent calculation of style ratings as well as in decisions pertaining to the creation of new gridletters.

At the beginning of each run, the Thematic Focus is empty, and it is gradually filled and modified as more examples of the gridfont are used to form the basis of the goal style. It tends to form a pyramidal shape, with a style being defined in terms of a very large number of stylistic properties that are relatively unimportant to the style and a few stylistic properties that are considered important to the style. A filled Thematic Focus may have well over one hundred stylistic properties in it, although typically somewhere between zero and two of them are on the highest level. A style is thus typically a very complex thing, although a brief list of a couple of traits can sometimes suffice as an approximate description of it. We believe that this kind of representation of style is consistent with the observation that people can readily recognize a style when they see it, but cannot easily define the style explicitly.

### The Library

The style of a gridfont is also represented with the help of the *Library*, a repository of all the gridletters and role-fillers that Letter Spirit has found among the seeds and among its own creations that have been used to define the style. The shapes in the Library can be useful in defining a style, both in creating new gridletters and in calculating ratings of style membership. This will be discussed in greater detail below.

## Examiner

The Examiner determines the letter category of an input gridletter. It was implemented by Gary McGraw as a stand-alone program, and can be discussed as such, although its intended and current purpose as to serve as a module within Letter Spirit. In addition to categorizing the input as an 'a', or a 'b', etc., it also segments the input into constituent parts and identifying for which abstract roles those parts are meant to fill. Finally, it returns a numeric rating of how well the input serves a member of its perceived letter category.

Processing in the Examiner follows an opportunistic course which is initially bottom-up, but seeks to exploit any information gained to the maximal benefit in guiding future processing. A run that accepted a relatively plain 'b' as input might proceed as follows: First, a Gestalt codelet would activate nodes in the Conceptual Memory letter categories based upon the coarse shape properties of the whole letterform; for a 'b', the Gestalt codelet would typically activate 'b'. This would lead to an attempt to segment the whole letterform into parts that are appropriate for 'b', namely a post on the left side distinguished from a bowl on the right side. Activation in Conceptual Memory would spread from 'b' to the nodes for 'left-post' and 'right-bowl'. After the program has attached descriptive labels (such as 'tall', 'thin') to each part, a Sparker codelet would attempt to bind a part to a corresponding role in memory, based upon matches between the part's labels and the role's permanent definition, but also favoring those roles that are more highly activated. Typically, a very simple letterform will be recognized in just a few dozen codelets. When early attempts have failed, codelets will very likely run that attempt to glue parts together or break bigger parts in two, seeking a segmentation that leads to successful recognition. After hundreds of codelets have run, the standards by which a part is expected to match a role are gradually loosened, allowing more and more exotic parts to be seen as members of less obvious roles. Atypical gridletters are thus often recognized after more time has gone by, when the right combination of segmentation and, if necessary, role loosening, has taken place.

The Examiner has received more development effort than the other portions of Letter Spirit combined. Logically, its performance is most crucial to the program as a whole (it is not possible to know what a fancy 'g' is without knowing what a 'g' is). It is has been tested as a stand-alone gridletter recognizer, and the latest version of the Examiner achieves 93.5% correct categorization on a large test set for which literate human subjects without special expertise in typefaces were only 84.0% correct.

## Adjudicator

The Adjudicator performs three tasks. First, it fills the Workspace with a representation, in terms of stylistic properties, of the style of a given gridletter. Second, and concurrent with the first task, it finds ways in which the Workspace would modify the Thematic Focus, should the gridletter be accepted as a member of the gridfont and worthy of being part of the definition of the goal style. Third, it uses the degree of fit between the Workspace and the Thematic Focus to calculate a style goodness rating for the gridletter.

The Adjudicator's codelet-based implementation differs from that of the Examiner in that the Examiner's processing often involves emergent loops, as one gridletter segmentation is attempted, and then rejected if it does not lead to successful recognition. The Adjudicator, in contrast, runs more like a straight-line program. Although the choice its codelets, which detect a variety of stylistic properties, is still nondeterministic, the course of a run is not open-ended. A fixed number of style-detecting codelets runs in nondeterministic sequence, building up a letterform-style structure that may differ somewhat from one run to another. The Adjudicator serves as a style evaluator as well as a style builder, producing a numeric rating of how well a single gridletter's style matches that of a gridfont as a whole. It can build up a style from many gridletters by accumulating properties from the set, promoting more frequently occurring ones to levels of greater significance in the Thematic Focus.

The Adjudicator depends crucially upon the Examiner, and expects as input the kind of output that the Examiner provides, not just a mere gridletter, but the gridletter annotated, as it were, in terms of letter category, parts, and roles. Thus, when the Adjudicator judges the style of a gridletter, it judges it *as* a 'b', or an 'x', or a 't'. Key aspects of the Examiner's design reflect that it is not a mere Optical Character Recognition program, but a servant to the Adjudicator.

## Drafter

The Drafter's task is to create a gridletter; the interaction of abstract roles and the concrete grid lead to a three-level hierarchy of activity to consider:

(I) Drafting a gridletter.
(II) Drafting a role-filler.
(III) Drafting a single quantum.

For the Drafter to render a gridletter, it must act on the top level. There are two ways to go about this. One is to create the gridletter in a single step, by borrowing and adapting a letterform from the Library. For example, an attempt at 'd' can be rendered by taking the mirror image of the 'b' (if there is one) that is part of that style. It should be stressed that this is not an infallible method of creating a good 'd'; for some styles, the mirror image of 'b' is **not** a good 'd' in that style.

The alternative to that is to create the gridletter one role-filler at a time. At that level, the Drafter may borrow a role-filler wholesale from the Library, or it may create the role-filler quantum by quantum. The drafting of a quantum is carried out by a run of the Coderack, which selects the next quantum for the role-filler out of all the eligible candidates. In this mode of the Drafter, role-fillers are drawn one quantum at a time, via successive point-to-point moves, as though an ink pen were doing the drawing. A run of the Drafter begins with the pen in some location, with various amounts of information on where and why it should draw next, and the output of one run of the Drafter is to decide which quantum, if any, should be drafted next. These coderack runs consult the Thematic Focus, as built up previously by the Adjudicator, and also the platonic definition of the intended letter category. The selection of each quantum can be thought of as a democratic process,

with each codelet voting for the penstroke that most suits its cause, which might be for one codelet the continuation of a motif that is common to the typeface; for another, making tall a part that whose role calls for tallness; for another, making the part shorter because the style calls for shortness. Codelets can and do conflict with one another. The eventual output must agreeably balance all constraints. The Drafter is in this way a good example of how the FARG architecture can avoid the brittleness that is so often a feature of AI.

The Drafter's output is not uniformly good. In fact, many of its attempts at a gridletter are poor versions of the intended letter category, or style, or both. Some attempts are even completely unrecognizable. This does not weaken the Letter Spirit program because no gridletter is accepted as a final member of the gridfont without the approval of both the Examiner and the Adjudicator.

### The top-level loop

The Examiner, Adjudicator, and Drafter are each programs of considerable sophistication. The top-level control of Letter Spirit ties them together into a relatively detailed model of human creativity in the domain of gridfont design. Its task is to carry out "the central feedback loop of creativity", in which everything the program creates is inspected for quality before it is deemed worthy of inclusion in the final output. A second objective that occurs simultaneously (in some versions of the program) is the evolution of a new style, in which new additions to the output can change the goal style that governs future output.

The Letter Spirit top-level program consists of two phases:

The first phase uses the seed letters given to the program to create a representation of the style that they have in common. The second phase of the program is a loop, in which a letter category is selected (nondeterministically, but favoring those categories that do not yet have a good version of them in the gridfont), and then the Drafter renders a gridletter that, ideally, incorporates the goal style as well as that letter. The Drafter's attempt is run past the Examiner and the Adjudicator, and if the attempt is the best version thus far for that category, as determined by the scores that the Examiner and the Adjudicator generate, then it is kept as the current version of that category in the gridfont. This loop runs many times, and as it does so, the quality of the gridfont should incrementally increase.

The top-level program of Letter Spirit lacks most of the characteristics of the FARG architecture, and is, in fact, a relatively simple program. The simplicity serves a number of causes. First, this makes it a relatively clean test of the strategy of the central feedback loop of creativity. Second, as a model of higher-level activity, it requires less in the way of fine-grained detail to model humanlike activity. And, of course, it is easier to begin with a simple implementation and add features and functionality in future work. A thorough exploration of Letter Spirit as it stands now is a valuable precursor to any future work that will increase the sophistication of the top-level program.

## Results

Letter Spirit and its constituent modules have each been tested in a variety of circumstances. For the sake of brevity, the only results that will be reported here are those of the whole Letter Spirit program. These runs each began with the program receiving five gridletters (in each case, 'b', 'c', 'e', 'f', and 'g') taken from one of five different human-designed gridfonts. Then the program strove to finish each gridfont by designing the remaining gridletters. In these runs, the program used up to 300 Drafter runs to generate multiple (a mean of 14) attempts for the 21 non-seed categories, keeping the attempt it rated best in each case. The resulting gridfonts appear in Figure 1.

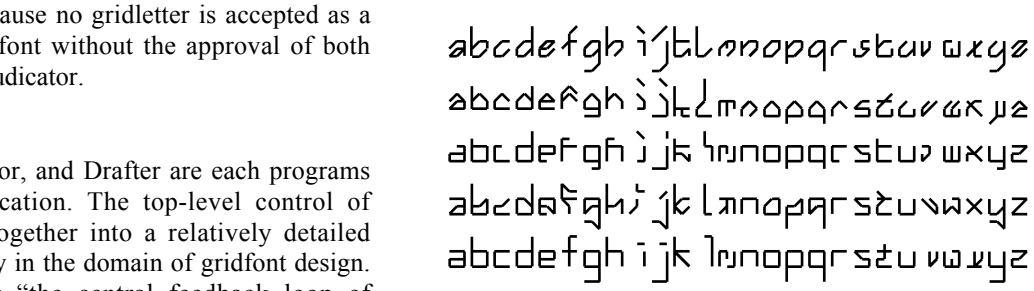

Figure 1: Five Letter Spirit-designed gridfonts.

### Observations

The gridfonts are both relatively legible and relatively coherent stylistically. Out of the non-seed gridletters (105 in all), all but perhaps five are easily recognized as members of their intended letter category. By scanning across the figure, one can easily see the general stylistic trend of each gridfont, although each gridfont has a few members that show little, if any, of the intended style. Approximately 30% of the gridletters are the exact ones used by the human designer in the original gridfont. Many of the remaining ones are only slightly different, or reflect the intended style despite going about the matter in a manner different than that of the original designer. In other cases, such as the tall 'd' in the third gridfont of Figure 1, which otherwise features short ascenders and descenders, or the squared-off 'w' of the first gridfont, one sees inconsistency in the program; it is not the case that Letter Spirit always does good work, but that it is capable of doing so. Fine examples of incorporation of letter with style are seen in the 'z' of the first gridfont, the 'v' and the 'x' of the second gridfont. Letter Spirit output has been exhibited in art shows (Rehling and Hofstadter, 2000) and, subjectively speaking, is of moderate, although not excellent, quality.

### Essential aspects of this approach

Letter Spirit has commonalities with other approaches in cognitive modeling, but diverges from them in important ways. For example, the Coderack has certain computational similarities to the way that production rules are implemented in architectures like Soar and ACT-R. Also, the review-and-revision strategy is a species of the widespread and more general generate-and-test approach that is ubiquitous in computer programming of all sorts.

However, it is important to note key aspects of the Letter Spirit program without which these mechanisms would be incapable of producing aesthetically plausible output. First, the nondeterministic production style of the Drafter is essential to the success of the review-and-revision approach. Simply put, it is of no benefit to produce multiple attempts at a product if all of those many attempts are identical. Second, the subcognitive level of codelets in all three Letter Spirit modules crucially distinguishes this work from cognitive modeling in the symbolic tradition. In order to display purposeful variability in behavior at one level of processing, a model must concern itself with a lower level. In the case of typeface design, there can be no rules at the role-filler level that instruct the model in definitive terms how to make tradeoffs between letter and style. Instead, the model must allow these pressures to compete with one another, allowing a range of possible results in the output, because it is not predictable a priori which influences should prevail in any given situation.

While it is usually the case that mature work in cognitive modeling includes a quantitative comparison between human and model data for the sake of partial validation of the approach, we have no intention of undertaking such an analysis here. It is *possible* in the case of models of aesthetic creation to perform strained versions of the comparison-with-subjects exercise but it is, we feel, for numerous reasons not only beside the point but also misleading. The goal of this work was not to implement most or all of the complexity of human-level creativity but rather to identify mechanisms that are sufficient to enable a semblance of creative behavior and to provide a case study that allows one to begin to consider which of those mechanisms are necessary. Any effort to quantitatively assess progress would derive more from the statistical methods used than it did from the model. It is certainly not a coincidence that artistic creativity is an endeavor that is hard or undesirable to evaluate quantitatively and that it has been less the subject of modeling work than human behavior that *is* easily captured quantitatively. Members of the artistic community itself have argued vehemently, if not always with logic, that quantitative expressions of aesthetic quality are misguided (Rehling, 2000), and we would argue that adhering to the usual standards of empirical validation is actually a hindrance to the enterprise of modeling creative behavior.

### Analysis

Letter Spirit produces moderately impressive output thanks to several positive aspects in its implementation, described in greater detail than is possible here in (McGraw, 1995) and (Rehling, 2000). One strength is the cognitive plausibility of its modules, particularly that of the Examiner. Another is the review-and-revision strategy of creativity, an approach that is probably hard to beat in terms of performance with one-pass approaches that do not review their own output and hope to create high-quality output on their first and only try.

Letter Spirit also has some clear shortcomings that could be addressed in future work. First, the Drafter completely finishes drafting an attempt of a gridletter before that gridletter is evaluated for quality (by the other modules). This allows the program to draft blindly and leads to Drafter output being clearly inferior to human-designed gridletters. A second problem is that gridletters that are rated poorly by the Examiner and Adjudicator are simply discarded, so that the Drafter must begin its next attempt at that category from scratch. A human designer can modify flawed products, correcting their flaws to make a good final product. Future work should allow Letter Spirit to review-and-revise on the role-filler level, as well as on the letter level. A third problem is that style needs to be represented in richer and more flexible ways. It is hoped that this work's strengths will motivate future work on this project and projects applying the same architecture to other domains.

## References


Erman, L., Hayes-Roth, F., Lesser, V., and Raj Reddy, D. (1980). The Hearsay-II Speech-understanding System: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213-253.

French, R. (1992). *Tabletop: An emergent stochastic computer model of analogy-making*. PhD thesis, University of Michigan, Ann Arbor, Michigan.

Grebert, I., Stork, D., Keesing, R., and Mims, S. (1992). Connectionist generalization for production: An example from GridFont. *Neural Networks*, 5.

Hofstadter, D., and the members of FARG (1995). *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought.* New York: Basic Books.

McClelland, J., and Rumelhart, D. (1981). An interactive-activation model of context effects in letter perception: Part 1, an account of basic findings. *Psychological Review*, 88(5):375-407.

McGraw, G. (1995). *Letter Spirit (part one): Emergent high-level perception of letters using fluid concepts*. PhD thesis, Indiana University, Bloomington, Indiana.

Marshall, J. (1999). *Metacar: A self-watching cognitive architecture for analogy-making and high-level perception*. PhD thesis, Indiana University, Bloomington, Indiana.

Mitchell, M. (1993). *Analogy-making as Perception.* Cambridge, Massachusetts: MIT Press/Bradford Books.

Rehling, J. (2000). *Letter Spirit (part two): Modeling creativity in a visual domain*. PhD thesis, Indiana University, Bloomington, Indiana.

Rehling, J. and Hofstadter, D. (1997). The parallel terraced scan: An optimization for an agent-oriented architecture. *Proceedings, IEEE First International Conference on Intelligent Processing Systems*.

Rehling, J. and Hofstadter, D. (2000). Letter Spirit: A model of creativity in a visual domain. *Art show at the XVI Congress of the International Association of Empirical Aesthetics*.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1--23.