Letter Spirit:

Recognition and Creation of Letterforms Based on Fluid Concepts

Gary McGraw

Center for Research on Concepts and Cognition
Department of Computer Science
Indiana University, Bloomington, Indiana 47405
gem@cogsci.indiana.edu

June 11, 1992

1 Creativity and Artificial Intelligence

The Letter Spirit project is an attempt to model central aspects of human creativity on a computer. Since it is the flexible and context-sensitive nature of concepts that allows creativity to flourish in the human mind, concepts are a major focus of the Letter Spirit research. Douglas Hofstadter has coined the term *fluid concepts* in order to reflect the essential plasticity of concepts [Hofstadter, 1987]. At the Center for Research on Concepts and Cognition (CRCC) we use computer-modeling to explore the nature of human concepts and their blurry-edged associative structure. We believe that creativity is a by-product of the fluidity of concepts, and that a reasonable model of conceptual fluidity can shed much light on creativity.

Letter Spirit is concerned with particular sorts of fluid concepts — those underlying letterforms. The aim of Letter Spirit is to model how the 26 lowercase letters of the roman alphabet can be rendered in a uniform style. The program will start with one or more seed letters representing a style, and create the rest of the letters in such a way that they share the same style, or *spirit*. Letter Spirit involves a subtle blend of high-level perception[1] and conceptual play that will allow it to *create* in all the interesting senses of the word. Letter Spirit explores creativity through the use of a design task, the art of letter design. Like most visual design tasks, letter design is deeply tied to the conceptual level.

Artificial Intelligence (AI) has not paid much attention to creativity. Most of the work that has been done (e.g., [Nanard et al., 1986], [Hammond, 1989], [Schank and Leake, 1989], [McCorduck, 1991], [Chamberlain, 1984], [Langley et al., 1987]) has focused on acts of creativity that are more *mechanical* than *conceptual*. Such programs usually tune superficial features in their domains or create "new" things that are actually hybrids by cutting, splicing, and recombining pre-packaged representational objects. The conceptual aspects of creativity have been largely ignored. The CHEF program, for example, creates a new recipe for stir-fry beef and broccoli by substituting "broccoli" for "green beans" in a recipe that it has for stir-fry beef with green beans [Hammond, 1989]. Though the program is itself goal-based (which means that its behavior is controlled at a fairly high level of abstraction), its "creative" aspect is largely mechanical.

There are two categories that most of the quasi-creative (mechanistic) AI programs mentioned above fall into: those that are tools for human creativity and those that implement syntactic substitution-and-manipulation schemes. Of course, these two categories are not mutually exclusive, so it is common to find a program that fits into both. We will provide examples of both types, tackling programs implementing syntactic transformation first.

[1] For a cogent explanation of high-level perception see [Chalmers, French, and Hofstadter, 1990].

Creativity and syntactic processing

Computers are very good at mundane, syntactically-based substitution (e.g., putting a 'G' wherever one finds a 'G'). Although the 'G' example borders on triviality, its message is really not too far off the mark. Syntactic transformations involve mindless mathematical manipulation and can be used either alone or in conjunction. Examples of simple syntactic transformations include: reflection, rotation, skewing, stretching, warping, expansion, compression, outlining, video reversal, adding a shadow, filling with a pattern, and substitution. These sorts of transformations, and possibly a few others, are usually included in programs like MacPaint or Idraw.

The DAFFODIL program discussed in [Nanard et al., 1986] uses very simple substitution methods to "create" letterforms. By replacing sections of a "letter skeleton" with human-designed complicated curvy lines, DAFFODIL generates some eye-catching, fancy letterforms. The problem is that the program itself is not doing anything interesting. *People* create both the letter skeletons and the fancy fillers — the computer simply substitutes a fancy filler for a section of skeleton according to its built-in rules.[2]

Some people have claimed that computers implementing the sort of simple substitution that DAFFODIL does are "creating new things" and should for this reason be considered creative. We do not believe that computers are doing anything creative when they implement syntactic substitution and/or manipulation alone.

We are especially skeptical of claims to creativity when a program's syntactic manipulation has a completely random flavor. If a program simply attempts all possible substitutions, haphazardly chugging away on a list of possibilities, its undirected syntactic approach leaves the interesting aspects of creativity to whoever has to pore over the output looking for the "good stuff." For example, even though they utilized powerful heuristic search strategies, Douglas Lenat's AM and Eurisko programs required a human perceiver to filter their results in order to find and remove the bad ones [Lenat, 1982][Lenat, 1983]. Although AM and Eurisko "discovered" some mathematical theorems, they had help in high places.

Another problem with blind manipulation is that it can easily lead to combinatorial explosion, as Chuck Rieger's work on syntactic, match-driven inference chaining clearly shows [Rieger, 1975]. Rieger's program blindly chained together inferences in order to build connections between propositions, and was quickly overwhelmed by the explosion of possibilities. It is clear that syntactic processing requires guidance from a higher level of abstraction — the level of concepts and goals. Processing under high-level conceptual guidance is known in AI circles as *top-down* processing.

We believe that creativity requires more than a blind syntactic approach. It requires a perceptual system that ties into a set of concepts, and that can therefore

[2] More about the DAFFODIL program can be found in section 5.1.1.

guide substitution and manipulation (or syntactic play) in intelligent ways and, more importantly, judge the results. In order to correctly model creativity in AI, a perceiver/judge must be built into the program [Hofstadter, 1985a]. Using human help is cheating. Letter Spirit will rely heavily on processing at the level of concepts. Its conceptual network, made up of a many interrelated concepts with associative links between them, will guide its creative behavior.

Tools for human creativity

Some researchers have written programs that are valuable creativity-enhancing tools; however, these programs do not *create* by themselves — they are merely tools designed to enhance the creativity of their users. The Meta-Font program by Donald Knuth is a perfect example [Knuth, 1982]. In Meta-Font, all critical perceptual tasks (including judging the final result) are left to the human user. The program simply provides a nice set of knobs to twiddle. Meta-Font serves as a powerful design tool, but it no more creates a font than a power saw builds a house.

All of the programs mentioned above (Aaron, AM, BACON, CHEF, DAFFODIL, Eurisko, Meta-Font, and RACTER) and others like them lack the same critical feature in their attempts at computerizing creativity — a foundation in high-level perception. We feel that high-level perception ("the process of making sense of complex data at an abstract conceptual level"[3]) is a crucial feature of all creative activity. High-level perception involves processing information to the level of *meaning* by accessing concepts and making sense of sensory data at a *conceptual level*. The malleability of concepts and their interwoven, associative nature allows creativity to flourish in the human mind. Creative acts, at their most interesting, involve conceptual play.

In human creativity, syntactic play is directed by top-down conceptual pressures (i.e., ideas). Casting creativity in terms of search through a space of possible creations, we can see that creative search is neither haphazard nor unguided. Instead, it is guided by ideas that inhabit the conceptual level of our minds. The idea (or set of ideas) behind any creative play, drive it in what seem to be fertile directions. The conceptual level also plays a critical role in the *judgment* of a creation. In order to judge a creation, to see whether it fills the role it was intended to fill, a creative agent must understand what the end result of creative activity is supposed to be like. Such understanding takes place at the conceptual level. A strong yet flexible conceptual foundation is required for both guiding creative search and judging the results.

[3]From the abstract of [Chalmers, French, and Hofstadter, 1990].

Figure 1: What is it that all these 'a's have in common? What is the *idea* behind lowercase 'a'?

2 The Motivation of Letter Spirit

2.1 Letters as concepts

Letter Spirit focuses on a fundamental question of cognitive science and artificial intelligence: *What is a concept?* One specific form this question takes in the Letter Spirit research is: *What is the letter* 'a'*?* Letterforms are subtler than people generally realize. Since most of us can read, we tend to take letters for granted, not considering them deeply. Most people think the letter 'a', for example, is just a *shape*. A closer look reveals an interconnected web of concepts that make up the idea of 'a' itself. Consider, for instance, how 'a' can be thought of as a marriage of two parts: a small c-like loop at the bottom, and a curvy umbrella-handle-type bar on the right. These two parts are not explicit shapes *per se* but are *ideas* for what the shapes eventually drawn should be like, what their acceptable bounds are, and how far they can be stretched before they no longer work. Often the drawn shapes deviate from the norm, stretching the flexible part-ideas in interesting ways. Figure 1 shows a number

of different lowercase 'a's. There is a common *idea* behind all the 'a's in Figure 1, but not a common prototypical shape.

Letter categories have flexible but strong boundaries that separate members from non-members. Some of the 'a's in Figure 1 are better than others. The weaker ones are found near the boundaries of the concept 'a', and help to define what being an 'a' is. Boundary lines between categories are flexible to the extent that they shift under contextual pressures. What is clearly a non-'a' in some situations may be a passable 'a' in others.

In order to better tease the concept of a letter apart from its geometric shape, we must introduce some new terminology. We will use the term *letter-concept* to refer to the abstract idea for drawing a letter. A letter-concept is present in a designer's mind before any marks are put on paper. The actual shape drawn on paper will be called a *letterform*. This distinction is important since Letter Spirit is concerned with playing with letter-concepts and instantiating the resulting concepts as letterforms.

Letterform design is an intricate and beautiful art making use of a large number of *letter-concepts*. Visual arts usually involve some type of conceptual design that goes on before any physical drawings or models are made. Many architecture schools offer entire classes on design conceptualization [Vickery, 1983]. The key to creativity, we argue, lies in understanding how this conceptualization works.

One way to get a handle on the shape/concept distinction is to consider lowercase 'x'. For most US-educated adults, the letter-concept for 'x' consists of a forward slash and a backward slash of the same size that cross somewhere near the middle. It is critical to stress that what is in people's minds is not a *picture* of two lines, but a set of *ideas*. However, since there is a very tight coupling between the ideas and their actual visual counterparts, it is difficult to pry them apart. There *is* a distinction, though, and comprehending it is necessary in order to fully grasp the Letter Spirit project.

English schoolchildren, in contrast to Americans, are taught to draw a lowercase cursive 'x' as two small crescents facing away from each other but "kissing" in the middle. If we look at a printed 'x' in this way, we are suddenly struck by this new conceptualization. The shape drawn on paper may be the same, but the conceptualization of it — the letter-concept in our mind — is very different.

Of course there are many ways of conceptualizing lowercase 'x'. We have mentioned only two above, but there are more: 'x' could be thought of as a lambda with a small forward slash on the top right, or as a 'v' sitting on top of an upside-down 'v', and so on. The question is, are some of these conceptualizations better than others? Is one of them the *right* way to think about 'x'?

The answer is that some conceptualizations *are* better than others. This is because the letter-concept one has in mind during the design process will strongly influence the letterform that ends up on paper. Different letter-concepts, even within the

same letter-category, lead to radically divergent letterforms precisely because different letter-concepts tend to stretch and slip in distinct ways. If we let a bizarre letter-concept influence the design of a letter too much, we may end up with a shape so strange or far-out that it no longer fits in the letter-category in which it was meant to fit. Consider the letter 'x' again. If we conceptualize 'x' as a normal 'v' on top of an inverted 'v' but allow the joining of the 'v's to slip so that the top 'v' moves down the right side of the bottom 'v', the conceptualization may drift out of 'x' space and toward 'n' space. This undesired sort of slippage would be less likely to occur under the standard "crossing slashes" 'x' conceptualization.

A given letter-concept may have to be slipped (i.e., changed in reasonable ways under conceptual pressures in order to relieve tensions) so that it will fit stylistic tendencies. It may even have to be radically slipped. Distinct letter-concepts (even within a given letter category) will tend to slip in different ways. Some might be better cut out for the stylistic pressures at hand than are others. For this reason it is critical to be able to reconceptualize a letter-concept during the design process.

To the extent that a letter-concept can be reshaped under contextual pressures, it is *fluid*. As we have seen, there are many possible ways to conceive of a given letter. Sometimes the possibility of reconception (slipping an entire letter-concept) is important in letter design. This is especially true when stylistic constraints are so strong that there is no easy way to design the letter given the current conceptualization. For example, under certain stylistic pressures it may be far easier to design an 'x' under the "kissing crescents" conceptualization than under the "crossing slashes" conceptualization.

In letter design there is much give-and-take between stylistic pressures and letter-category pressures. We characterize this give-and-take as a fight between two forces: an *attractive* force that tends to pull the shape being created as close to the center of the intended letter-category as possible (so that it is a strong instance of the letter), and a *repulsive* force that tends to push the shape away from the center of the letter category (so that it is a strong instance of the desired style). The conflict between these two forces lies at the heart of the Letter Spirit project.

2.2 Letter design as cognitive science

Letter perception and creation provide an elegant window on the workings of the mind. If we can gain insight into the workings of concepts and their fluid interrelations, we will have made important headway on the problem of intelligence.

Like music or writing, letter design is an art form that requires years of practice. Within Letter Spirit we hope to preserve the *deep* aspects of the art of letter design while removing some of the shallower artistic qualities that, for the most part, rely on well-practiced, refined motor skill, such as: use of elegant curves and tapering lines,

knowledge of typographic conventions, and the need for a well-trained hand. As an end result we are left with very stripped-down letters, which are much closer to the concepts behind them than fully fleshed-out letterforms.

As Hofstadter pointed out in [Hofstadter, 1985b], AI has had little to say on a deep level about letters. There has been a fair amount of work on letter recognition, but it almost always takes a purely syntactic (geometric) approach. So far, attempts at letter recognition and creation in AI have been based on the notion that letters are basically syntactic variants of a single underlying shape. We argue that one of the major problems of such an approach is that it ignores the philosophically interesting questions like: *What is* 'a'*-ness all about conceptually?* All 'a's share a common conceptual framework that is non-syntactic and non-geometrical in nature (as was argued above with reference to Figure 1). This framework and its use in letter design are what Letter Spirit aims to model.

A deeper question regarding letterforms is: *How are letters in a given style related to one another?* This can be cast in terms of an analogy problem. Given an 'a' in a certain style, how would you make an 'e' in the same style? The most difficult problem here is figuring out what "the same style" means. There is something extremely slippery here. Transferring stylistic aspects from one letter to another involves the sort of slippability discussed in [Hofstadter, 1979], [Hofstadter, 1987], [Mitchell, 1990],and [Hofstadter and Mitchell, 1991]. Stylistic qualities of one letter cannot usually be directly transferred to another; rather, they must be *slipped* into variants of the "same type" so that they fit within the conceptual framework of the new letter without bending it beyond recognition.

Letter Spirit is meant to address two important aspects of letterforms: the *conceptual* sameness possessed by letters belonging to a given category (e.g., 'a') and the *stylistic* sameness possessed by letters belonging to a given style (e.g., Helvetica). A simple picture illustrates these two ideas and shows their relationship to one another (see Figure 2). One of the two central problems of the Letter Spirit project is to characterize how members of entirely different categories can share "the same spirit". The other is to discover what being a member of a letter-category is all about. The idea is that given a few letters in a *gridfont* (our name for the highly constrained, gridbound alphabets used in the Letter Spirit project), the program "figures out" both what category a given seed letter belongs to, and what the stylistic tendencies suggested by the seed letters are, and then goes on to design the remaining letters in the same style. Of course, there is never just one good answer since seed letters do not completely specify a style, and a style does not completely specify its constituent letterforms.

Letter Spirit has multiple goals in that it is both a cognitive model and an AI program. Letter Spirit approaches the task of understanding creativity by way of cognitive modeling. One of the tenets of cognitive science (at least of our flavor) is

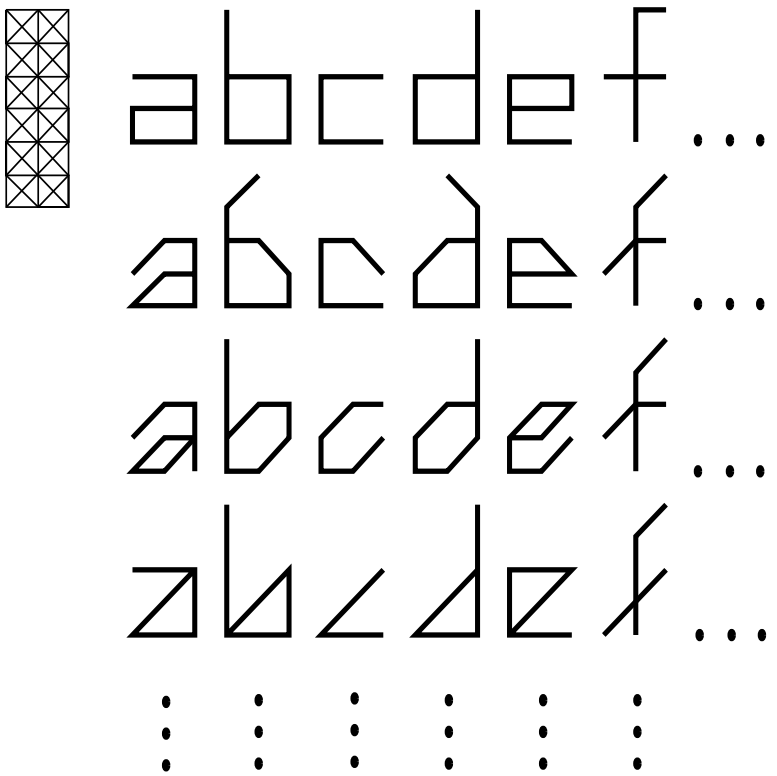

Figure 2: Items in any column have *Letter* in common. Items in any row have *Spirit* in common.

that models exhibiting reasonable approximations to psychological behavior will shed light on the mechanisms underlying that behavior. An important feature of cognitive modeling is the manipulation of a model in order to discover how it behaves. The Letter Spirit architecture is based on a particular model of human cognition in which *concepts* play a central role. It is hoped that Letter Spirit's fluid conceptual basis will give rise to creative behavior qualitatively similar to human behavior.

Letter Spirit is also an AI program. Even if its cognitive mechanisms turn out to be very different from human mechanisms, Letter Spirit will still be interesting as an approach to machine creativity. Symbolic AI has been criticized for its mechanistic brittleness. There is little doubt that brittle behavior is incompatible with creativity. A typical AI program would be hard pressed to exhibit creative activity. New mechanisms are needed if AI is to successfully model creativity (along with a host of other human activities that require intelligence). We believe that the Letter Spirit architecture, which is based on sub-cognitive, emergent computation, will extend the ability of computers to act intelligently.

3 The Domain

3.1 Typeface design

The initial idea that spawned the Letter Spirit project was to write a computer program to design full-fledged typefaces having the same order of complexity as conventional typefaces such as Helvetica, Times, Baskerville, and so on. This turned out to be far too hard. The problem is that real-world typeface design involves too many levels to allow practical simulation in a computer model.

Some of these levels are fairly superficial (e.g., figuring out the exact amount of curvature to use or the thickness of the marks to make) and involve syntactic manipulation, at which computers excel even without using AI techniques. In fact, some models using purely syntactic approaches to letter design have been produced [Nanard et al., 1986]. These models focus on the frills of letter design, and ignore its deep aspects. Consequently, they shed little light on creativity.

This is not to say that impressive things cannot be done using syntactic methods. All the 'a's shown in Figure 3(a) were created by syntactically manipulating an 'a' from the Times Roman font. The weird non-'a'-like shapes shown in Figure 3(b) were created by the same sorts of syntactic manipulation of a Times Roman 'a'. Certainly, syntactic manipulation is a very powerful tool — but it is mindless. Consequently, if it is to be used by a computer, it must be driven and controlled by top-down, conceptual pressure. Without monitoring and guiding at the conceptual-level, syntactic manipulation can push shapes far beyond acceptable letter-category bounds (as Figure 3(b) shows). There is no reason that syntactic play should be excluded from AI programs, but such programs must be sensitive to the effects of syntactic manipulation, and be able to use such sensitivity to control the type and amount of manipulation that gets done, so that results are reasonable. In other words, there must be some clear purpose or idea behind syntactic manipulation.

Some of the levels of typeface design are deep and involve playing with the concepts behind the visual shapes. We feel that by deliberately ignoring many of the surface-level aspects of typeface design, we can concentrate on the deepest, most *conceptual* levels of the design process. We are concerned more with what being an 'a' is all about than with designing beautiful but superficial 'a's. Our hope is that such a move will help us get to the essence of human creativity. By stripping the domain down to a minimal level we can avoid having to worry about the surface-level details and concentrate on the essentials.

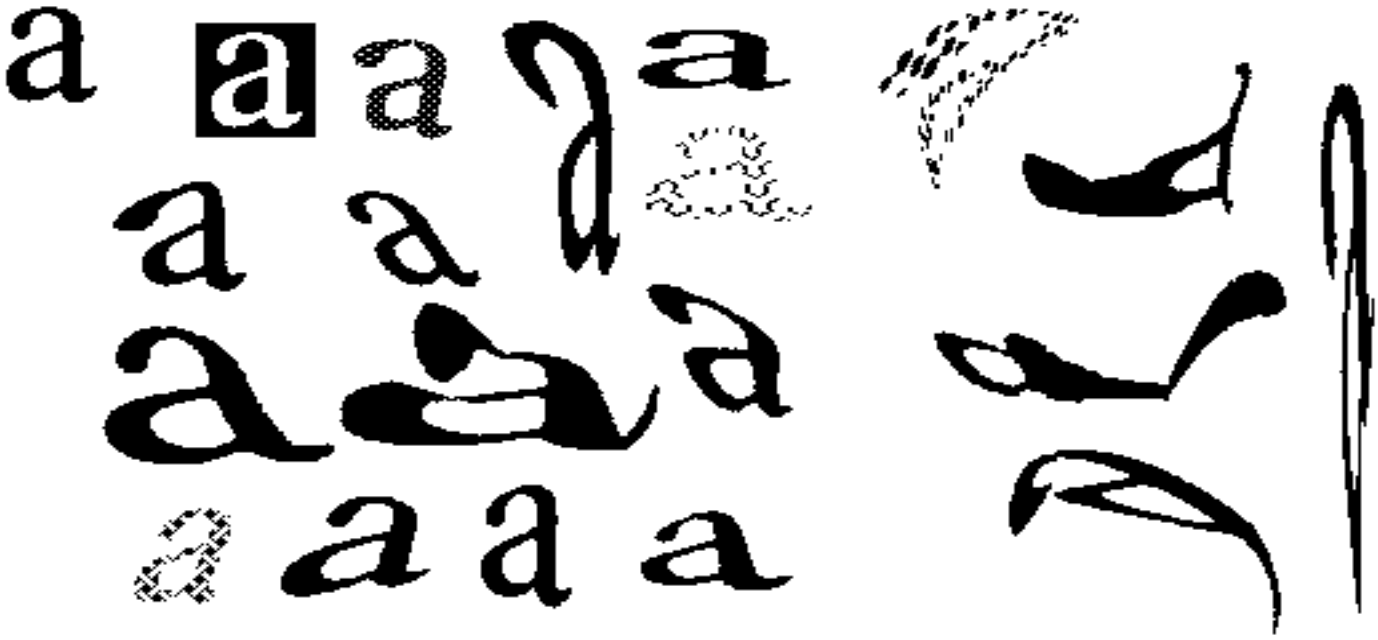

Figure 3: (a) On the left hand side of the picture are lots of 'a's created by syntactic transformation of the 'a' from the Times Roman typeface. The original 'a' is shown in the upper left corner.
(b) On the right are some non-'a' shapes created by exactly the same sorts of syntactic transformation of the Times Roman 'a'. Of course, it is possible to get even weirder shapes.

3.2 The grid

Thus the Letter Spirit grid was born. The grid has 21 points arranged in a 3x7 array. Legal line segments, called *quanta*, are those that connect any point to any of its nearest neighbors, horizontally, vertically, or diagonally. There are 56 possible quanta on the grid.

The grid can be divided into three zones corresponding to locations where parts of letters usually fall. The *central* zone is where the letters 'a', 'c', 'e', 'm', 'n', 'o', 'r', 's', 'u', 'v', 'w', 'x', and 'z' are found in their entirety. The other letters — 'b', 'd', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'p', 'q', 't', and 'y' — have ascenders or descenders that jut into one of the other two zones, called, naturally enough, the *ascender* and *descender* zones. The bottom of the central zone is called the *baseline*, and the top is called the *x-height* line. A labeled grid is shown in Figure 4.

Letters created on the grid can be seen either as self-sufficient letterforms in their own right or as "skeletal" geometric representations of letter-concepts that can be

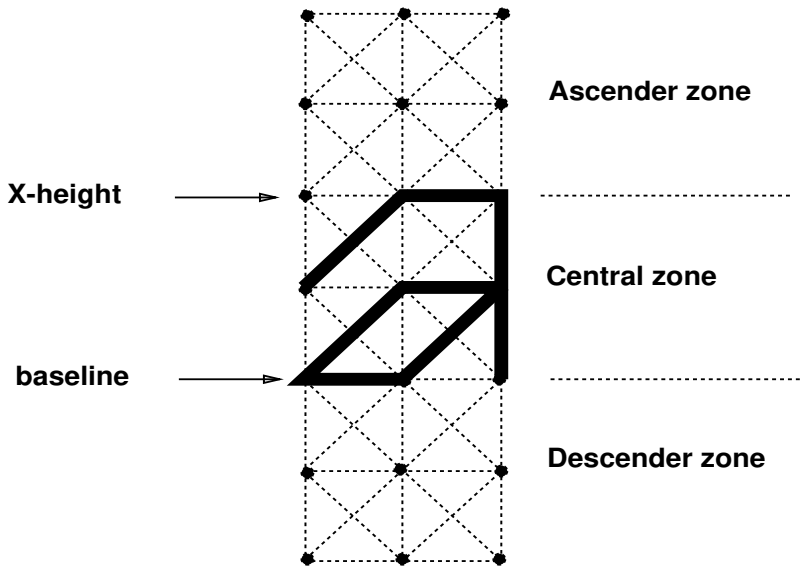

Figure 4: The grid.

"fleshed out" into true letterforms. Figure 5, for instance, shows the relationship between a gridfont called *Benzene Right* and a derivative "dressed up" alphabet. Using straightforward and totally mechanical shape-for-shape substitutions, any gridfont can be converted into a more conventional typeface. A chart of the substitutions involved is shown on the left side of Figure 5. We have chosen to leave out the step of adding "flesh" to the "bones" of a gridfont, and to work at the more basic level of the "skeletons". These skeletons may of course still be considered as geometrical representations of a given typeface's conceptual framework.

This is not to say that gridfonts should not be considered as stand-alone typefaces. Some are especially beautiful, and make acceptable typefaces; but most of their beauty flows from the ways in which they illustrate and explore the conceptual essences of letters. When style and letter interact, the boundaries of letter-concepts become extremely relevant.

Since quanta are strictly binary (i.e., are either on or off), decisions on the grid are coarse. Surprisingly, variety among letters of a given category is still huge. So far, about 600 gridfonts have been designed.[4] Surveying them all, one is struck by the diversity present in each letter category. Almost paradoxically, the gridfont domain's severe limitations seem to engender this diversity. The Letter Spirit grid's discreteness removes the possibility of continuous change and therefore forces a designer to tamper with the essence of a letter. In fact, since the grid is so coarse, decisions to add or

[4]Six completed gridfonts are included at the end of the paper.

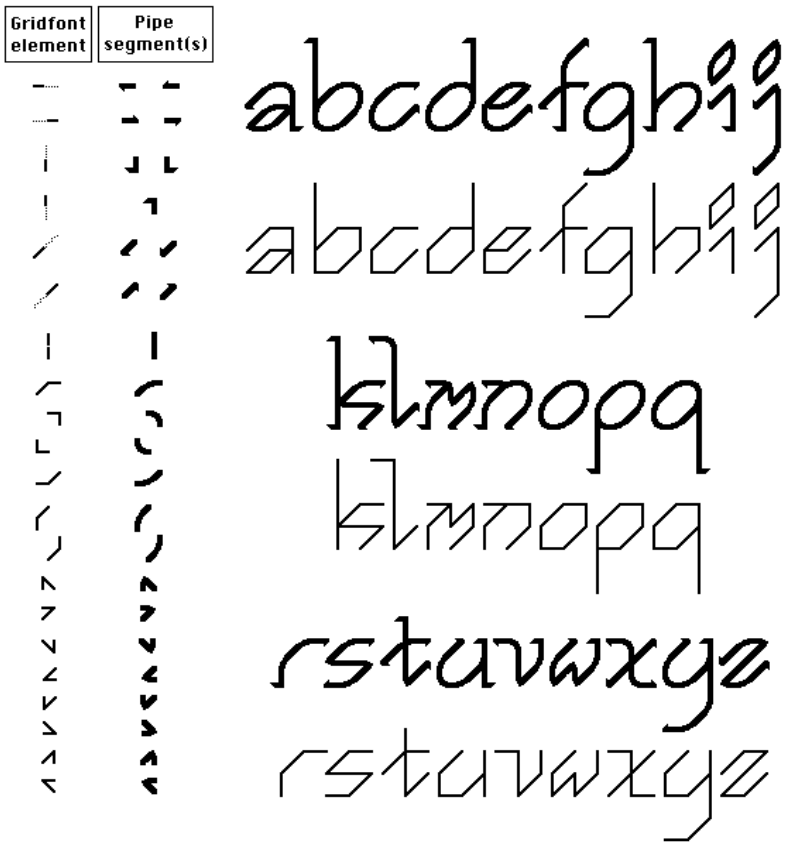

Figure 5: The gridfont Benzene Right is displayed below its fleshed-out pipeface cousin.

subtract even one quantum can fundamentally affect category membership.

As an example of the distance in letter-space one can travel with minute changes on the grid, consider Figure 6. A "small" change in the gridfont letterform — just one quantum — changes the letter from a strong 'z' to a strong 'a'. A different one-quantum change can transform the same 'z' into an 'e'. So category membership in the Letter Spirit domain is a tricky matter. In fact, deciding which of the 26 categories a given shape on the grid belongs to is one of the hardest problems of the entire project and represents the major focus of this thesis proposal.

The grid was invented in order to get to the heart of letters and styles as abstract concepts. We are interested in studying what we call *deep style*. Deep style involves stylistic elements at the conceptual level, while *shallow style* involves surface-level (usually geometric) feature manipulation. Shallow style is usually concerned with

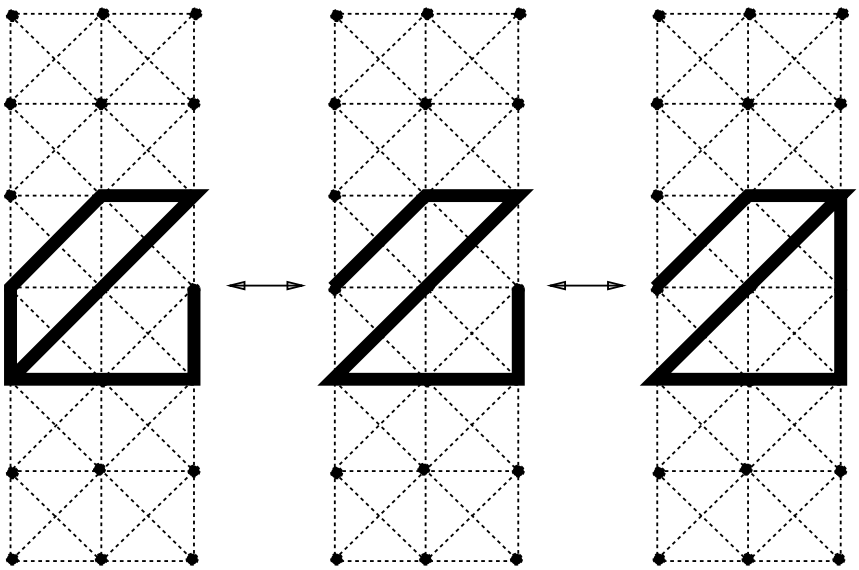

Figure 6: Metamorphosis of an 'e' into an 'a' through 'z' with a one-quantum change between each.

embellishment [5] or simple syntactic manipulation of the MacPaint variety.

We argue that this sort of surface play is not truly creative. Real style runs far deeper. At some point, mechanistic application of syntactic transformations no longer works on its own (i.e., transformations cannot be applied blindly without making a mess of things), and perception must be brought to bear to guide the transformation. This is where the conceptual level (in the form of top-down pressure or guided activity) is brought in, and style begins to get deeper. At the deepest levels, style involves the manipulation of the conceptual foundations of letters (e.g., conceptualizing an 'x' as "a normal 'v' on top of an inverted 'v' " instead of as "intersecting slashes" in order to relieve stylistic tension). The distinction between deep and shallow style is not a black and white one. Instead there is a continuum, with shallow style on one end and deep style on the other.

The gridfont *Intersect* (in Figure 7) shows how style can deeply affect the letterforms of a given gridfont. The main stylistic constraint in *Intersect* is that two curves must intersect to form each letter. This constraint is not hard and fast — it must be slipped in order to create acceptable versions of a few of the letters. Of course, style is not all that slips — letter-concepts must be stretched to fit within the style too (to the extent that such stretching is possible).

The Letter Spirit project involves coming up with an *idea* for a shape, then ren-

[5] The DAFFODIL program implements a type of shallow style manipulation [Nanard et al., 1986].

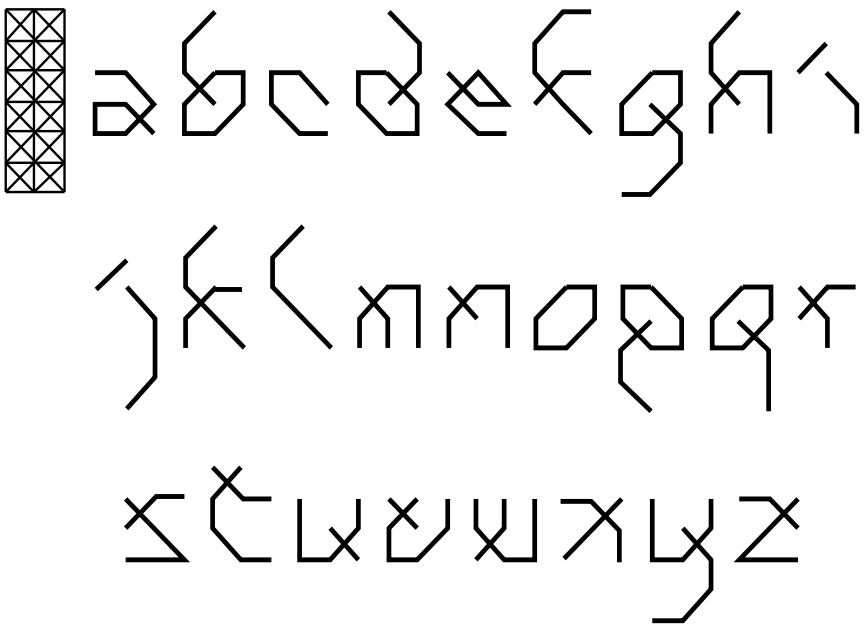

Figure 7: The gridfont *Intersect* illustrates how style can affect letterforms in the gridfont domain.

dering it physically on the grid. This rendering must be perceived and critiqued by a perceptual agent. In Letter Spirit, the program decides what is acceptable as an 'a' *in the current style being considered.* The essence of 'a'-ness is not completely specified by programmers in a static pre-packaged representation, rather it emerges as letterforms interact with the conceptual basis of the program. Instead of having 26 platonic *shapes* that get "fleshed out" with weird curves and other pre-designed objects (as in DAFFODIL), there are 26 *fluid abstractions*, one for each letter, which respond in a non-rigid way to stylistic pressures. Each such abstraction must be capable of adapting under pressure in such a way that its physical representative in the font being designed remains within the space of its intended letter-category. Letter Spirit has an abstract idea of what 'a' should be. This abstraction implicitly includes information about the outer limits of a letter category as well as information about the relationships between letter categories.

As we discovered when discussing the 'x' example above, letters can have many possible conceptualizations. Like the parsings of a Necker-cube, conceptualizations compete for perceptual validation. Only one of them may be consciously active at a time. Because there are rival conceptualizations, there are multiple frames within which a letterform can be "checked". If a shape seems to be both a good *normal on top of inverted 'v'* type 'x' and a good *intersecting slashes* type 'x', chances are it is

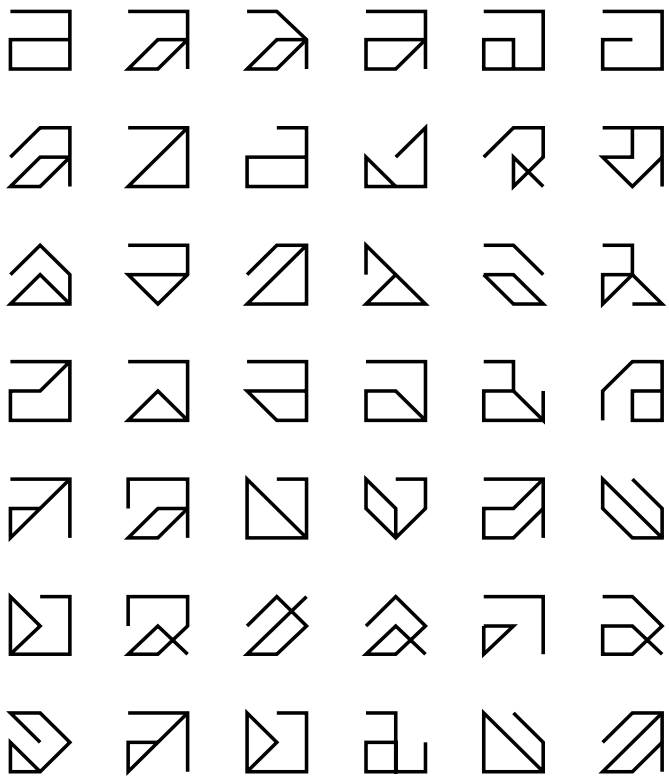

Figure 8: Several renditions of the letter 'a' on the grid.

very strong as an instance of 'x'. The more conceptualizations that the shape fits into, the better. As we mentioned above, some conceptualizations are better than others. These carry more weight in the checking process.

A new way of conceptualizing an 'x' can be validated by creating a few letterforms that fit the novel conception and checking to see how well they fit more standard conceptualizations. If the test letterforms are good instantiations of several different conceptualizations of 'x' then the new way of thinking about 'x' is probably reasonable.

On first encountering the grid, one may wonder whether the domain has been limited to such an extent that not much room is left for interesting letters. How could there possibly be more than a few styles on such a small grid? With around 600 gridfonts under our collective belt, we feel justified in saying that there is indeed much room for creativity. As an example of the richness the grid allows (even within one letter category), see Figure 8, which shows 42 of the approximately 1500 different 'a's that have been created.

As mentioned above, the constraints defined by the grid actually encourage tam-

pering at a deep level. Because there are no distracting surface features to daintily maneuver, it is common for one to end up playing at the boundaries of the 26 categories. For this reason many gridfonts are wild, sometimes having angular, spiky, sparse, contorted, and otherwise bizarre letters.

Most typefaces found in catalogs appear tame when compared with gridfonts. This is because they are created under far less severe stylistic constraints. In "vanilla" styles there is not much pressure to make style extremely conspicuous — the "repulsive" force mentioned above is weak. As a result the "attractive" force tends to hold a designer near the center of a letter category resulting in very normal looking letterforms. This is not to say that regular letter design is uninteresting or sterile — there is still considerable room for experimentation and play. But the play in normal letter design is less risky and wild than the play involved in designing a gridfont. Experimentation on the grid involves exploring the very nature of what it is to be a certain letter. This is what makes Letter Spirit intellectually challenging and stimulating.

Since gridfonts are a vehicle for creativity research, we need not worry ourselves too much over their inherent wildness. The wildness is a side-effect of a domain that forces deep-level — or, semantic — play. We are not striving for typefaces with maximum readability, or for letterforms that are beautiful at the surface level. Instead, we are attempting to understand the conceptual nature of letterforms. We thus eagerly push the 26 categories to their very edges, exploring the conceptual boundaries and their interactions as we go. As Jim Marshall, a fellow CRCC researcher, once said, "We love letters for their minds, not their bodies."

4 A Sketch of the Program

4.1 Creating a gridfont — a broad view

The creative process which Letter Spirit will model takes place over an extended period of time. For a person, designing a gridfont usually takes anywhere from ten minutes to three hours (after which the font is subject to continual scrutiny and minor revision). The entire process involves the combination of myriad small operations ranging from the inventive to the mechanical. Most actions set off repercussions that echo throughout the continuing design process. Waves of activity surge through the gridfont, washing over interrelated letterforms, causing change at all levels. Eventually the waves begin to settle down, and a stable gridfont emerges.

Dissatisfaction and contradiction either within a given letterform or between letters within a gridfont style are the prime motivators of the design process. If two or more letters seem to conflict stylistically then the resulting gridfont will be unsatisfying. Letter Spirit must resolve these conflicts in order to create a coherent

style. Sometimes the conflicts are non-obvious and tricky — requiring refined artistic judgement calls. Making Letter Spirit sensitive to trouble spots so that it can find where the trouble is, diagnose it, and find a reasonable solution, is a key aspect of the project.

Design decisions in Letter Spirit affect not only the letter currently under consideration but also many conceptually close letters in the gridfont. For example, a design decision regarding the "post" of a gridfont's 'b' (the line on the left side of a 'b') is likely to have enormous effect on the "post" of the 'd' as well as noticeable influence on the descenders of the 'p' and the 'q'. Certain aspects of letterforms, once they have been designed, must be propagated through the entire gridfont. The propagating wave will most likely cause many adjustments (both major and minor) to some of the "already finished" letters. It will also give rise to ideas for letters that have not yet been designed. One design decision will typically cause other design decisions to be made, which in turn engender others, and so on.

Eventually, when enough "decision waves" have washed over the entire gridfont, letterforms begin to have a high degree of internal consistency, and a consistent style begins to emerge across the entire gridfont. Once the tension of inconsistency eases up, no more large-scale changes are required. Minor adjustments will continue, but for the most part, the large-scale creative act will be finished.

4.2 Modularizing Letter Spirit

In order to make the rather large Letter Spirit project tractable as a thesis project (or two), we have divided the program into roughly two modules: the *recognizer-evaluator* and the *juggler-suggestor*. The recognizer-evaluator is intended to model letter perception (with emphasis on concept-level perception) beginning at some ill-defined stage of lower-level visual processing. The recognizer-evaluator decides which letter a given shape on the grid is meant to be. It also rates the "goodness" of the shape as a letter. The recognizer-evaluator must be able to evaluate letterforms both in terms of their fit within a given letter category and in terms of their fit within a particular style. It must decide whether the balance between letter and spirit in a proposed letterform is acceptable.

During a typical Letter Spirit run, a few seed letters, which start a new gridfont on its way, will be perceived and categorized by the recognizer-evaluator. The recognizer-evaluator will have to do much more than just figure out which letter a shape is — it will also need to make note of particular stylistic tendencies in the letterform and judge how good an instance of its letter category the letterform is. This is what sets the recognizer-evaluator apart from current approaches to letter perception. Stylistic tendencies will need to be made available to the juggler-suggestor, which will suggest new letters to add to the current gridfont. New letters will in turn cause further

stylistic reverberations to occur among the already existing letters of the emerging gridfont, perhaps changing existing letters' evaluations.

The juggler-suggestor module will suggest possible new letters for inclusion in the current gridfont. It will make its suggestions under stylistic constraints discovered by the recognizer-evaluator. Suggested letters will be evaluated by the recognizer-evaluator. Some letters may pass initial inspection, possibly causing waves of revision to course through the rest of the gridfont. Other suggested letters may be rejected by the recognizer-evaluator and will need to be modified by the juggler-suggestor. Any modification will have to be based on failure information provided by the recognizer-evaluator.

The recognizer-evaluator and the juggler-suggestor will work in tandem during the creation of a gridfont. The design process as a whole involves the interaction of perception and suggestion. Work on the gridfont will take place in a common memory structure similar to the *Blackboard* of Hearsay-II [Erman et al., 1980].

Since design decisions have multiple repercussions on the gridfont, many possible courses of action will be available to Letter Spirit. This is true for both within-letter processing and between-letter processing. Instead of completely centralizing control, multiple paths will to some extent be explored in parallel (with the most fruitful paths getting the most processing power). This is especially true of within-letter processing. The architectural mechanisms required for highly parallel computation of this sort are a key feature of Letter Spirit. They will be explained below.

5 The Recognizer-Evaluator

5.1 Processing in the recognizer-evaluator

My thesis work will concentrate on the recognizer-evaluator module. We may implement preliminary versions of the juggler-suggestor so that we can get an initial prototype of Letter Spirit working, but only the recognizer-evaluator module will be thoroughly and completely polished. Work on the recognizer-evaluator module will focus on high-level letter perception and evaluation, with special emphasis on the perception of style.

In order to understand the recognizer-evaluator process clearly, it is useful to cast it in terms of *syntactic* and *semantic* operations. Syntactic operations are purely bottom-up chunking operations. They serve to put quanta together in a way that would be reasonable no matter what type of shape was being recognized. They are completely *context-free* chunkings that presumably would arise as the result of any naturally-evolved visual system. Semantic operations, on the other hand, are *context-dependent*. They depend on the specific repertoire of categories into which the stimuli

are to be filtered — a repertoire that in people has been acquired through experience. Semantic operations take the output of syntactic actions (which usually occur earlier in perceptual processing) and adjust and refine it. They result in the "marriage" of sensory stimuli with concepts. The context-supplying categories of Letter Spirit are the 26 lowercase letters.

In brief, then, operations that depend in any way on the final set of high-level conceptual categories into which stimuli are to be classified are semantic. Low-level operations that are independent of those categories are syntactic.

Work on the recognizer-evaluator explores the gray area between syntax and semantics. High-level perception involves channeling initial sensory stimuli through a set of syntactic processes up into semantic space. The recognizer-evaluator will implement a model of recognition at the conceptual level. The idea is to model, in a rough sense, the later stages of visual perception whereby a shape is recognized as a particular letter.

Processing begins at the level of quanta and starts out being completely bottom-up. Quanta are syntactically chunked into parts, which are then labeled with low-level syntactic characteristics. Top-down semantic influence enters the picture as the labeled parts are matched up with what we call conceptual *roles*. Harking back to our versatile 'x', the two slashes in the "intersecting slashes" conceptualization are examples of such roles. Eventually, as the interpretation rises towards the level of letters (which we call *wholes*) even more top-down influence is brought to bear. As was mentioned above, letters have more than one possible conceptualization. Such conceptualizations (or letter-concepts) are represented as sets of interrelated roles. Standard conceptions of the 26 letters (usually one or two for each letter) are provided to the program as conceptual prototypes, but the program should be able to develop and test new conceptualizations during a run (as mentioned above).

The recognizer-evaluator operates roughly as follows (starting with quanta and moving towards classification of the shape as a letter). Be forewarned that the following outline may give the impression that processing is more serial and well-ordered than it really is. In reality, processing occurs in a more parallel manner, with different levels of processing often proceeding concurrently. The outline simply provides a simplified way of thinking about the recognizer-evaluator.

- Quanta are probabilistically bonded together (by local perceptual agents, called glue-ants) with different amounts of "glue". More glue is deposited (stochastically) in straight intersections than on angles. The glue-depositing glue-ants execute completely syntactic operations.

- When a fair amount of glue has been deposited, the "glommed" shape is metaphorically shaken. This amounts to probabilistically breaking the

glommed shape into parts at its weakest joints. Shaking breaks the structure into a set of chunks (usually made up two or three quanta) called *parts*. Since no concepts have been invoked up to this point, parts are still at the level of syntax.

- Each part is scanned by multiple syntactic agents that probabilistically attach *syntactic labels* to the part. Labels denote very simple properties of parts like curviness, length, width, and so on.

- The presence of a particular label on a given *part* serves as a cue that tends to lightly activate one or more *roles* with which the label is associated (i.e., roles of which the label is at least somewhat diagnostic). For instance, the labels "left-diagonal" and "straight" would tend to activate one of the "slash" roles of 'x'.

- Any role with sufficient activation will attempt to "mate" with the specific part whose labels activated it (here is where syntax and semantics meet). Roles compete with each other for a given part, vying to pair themselves off with parts. Usually the role that best matches a part will win a fight, but once again, role matching is stochastic. Of course this can all be viewed from the part's point of view as well, where parts are seen as competing for role attention. A useful metaphor for the part/role pairing is dating and marriage in human couples.

- As roles and parts attempt to couple, a given part may need to be slightly altered in order to make a good mate for a given role. A quantum or two may need to be stolen from one or more neighboring parts to make the part in question more attractive to a highly suited role. Likewise, small pieces that seem to make a part ugly in the eyes of possible role-mates may need to be given away or detached.

- Roles compete for parts throughout the letter, adjusting the part chunkings as they go. Presumably, when the adjustment phase is over, there will be a fairly strong match-up between roles and semantically adjusted parts.

- Each instantiated role will have a few comments attached, stating *how well* the given part instantiates the role. This information will include comments about deviation from norms associated with the role.

- Each realized role begins to alert one or more *wholes* (i.e., full letter-concepts such as 'a') for which it provides evidence. Role/whole coupling is

analogous to part/role coupling, only it occurs at a higher level. Particular letter conceptualizations will be chosen according to how well they match the instantiated roles.

- Different wholes become activated to different extents. Each sufficiently activated whole attempts to perceive the shape on the grid as an instance of itself. Wholes must take into account the commentaries attached to parts and roles.

- One of the wholes is picked as best. A list of stylistic features is attached to the letterform. Such features include both syntactic and semantic properties. Associated features will be used by the juggler-suggestor in suggesting instances of letter-concepts that have similar roles to the letter-concept just perceived.

The entire labeling and role-matching process will happen in parallel for each of the initial parts created after shaking. In general, processing proceeds from low-level syntactic processing to high-level semantic (or conceptual-level) processing, but is not completely linear in nature. Conflicting perceptual structures will compete against each other in the context of current perceptual trends. Bad structures will be destroyed and strong ones strengthened. All processing takes place in a global short-term memory area. This area implicitly contains information about the current state of processing, since all perceptual activity directly affects it. Numerous small stochastic agents running in parallel are the main computational elements of the Letter Spirit architecture. Their activities result in a globally coherent emergent perceptual structure.

Figure 9 shows how the recognizer-evaluator might recognize a 'b'. Some of the interrelations between levels of processing are abstracted away in order to make the picture more readable. It is important to remember that although the "roles" in Figure 9 are shown as shapes, they are actually more abstract conceptual entities.

5.2 Architecture of the recognizer-evaluator

Inspiration for the recognizer-evaluator architecture comes in part from a model of low-level and high-level auditory perception — the Hearsay II speech-understanding project [Erman et al., 1980]. Hearsay II modeled the way in which low-level auditory input signals are converted to meaningful utterances. It made use of a Blackboard system that integrated information from a variety of processing levels to keep track of processing state.

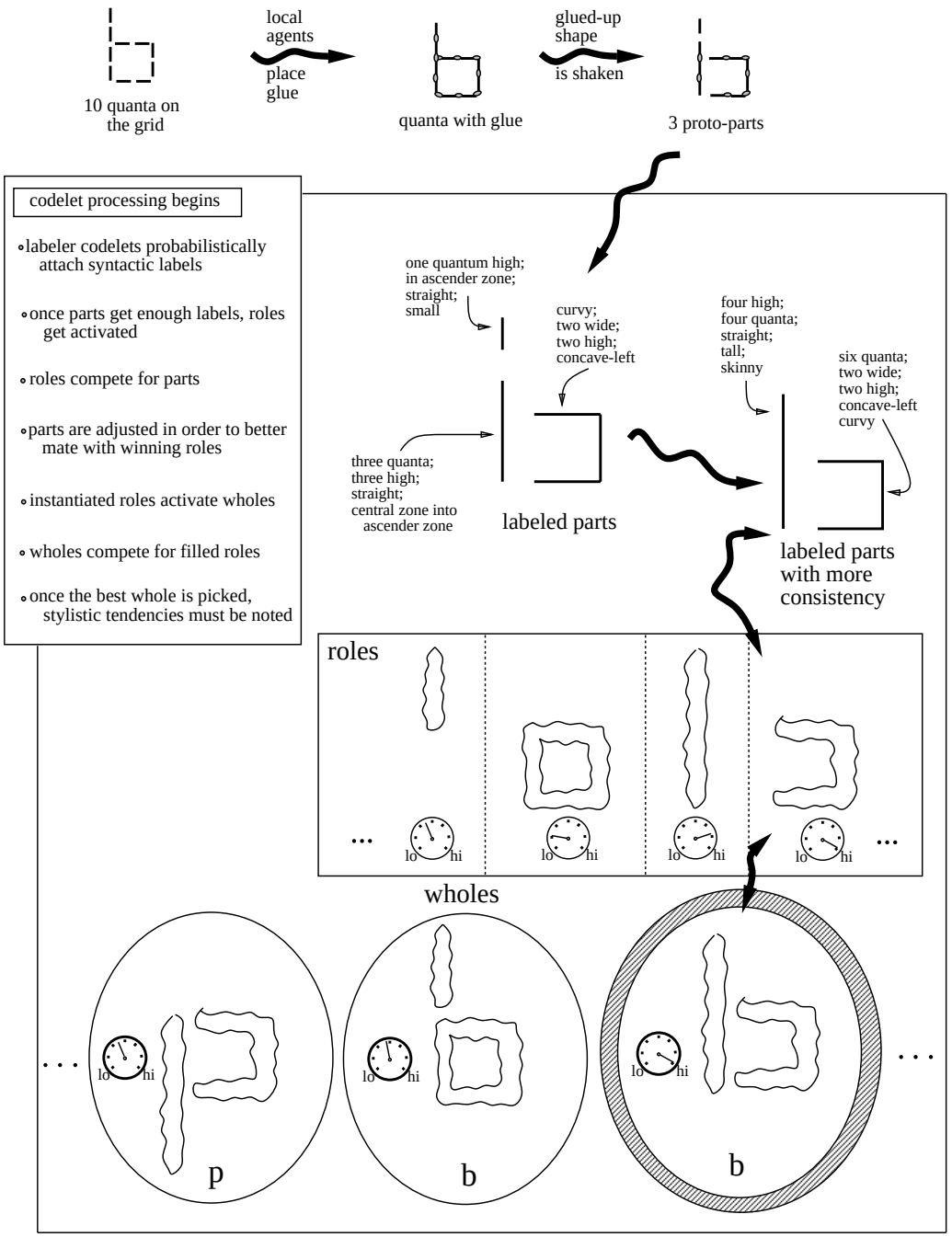

Figure 9: How a 'b' might be recognized by the recognizer-evaluator. The wiggly shapes at the bottom should not be thought of as shapes, but as roles.

The two main predecessors of Letter Spirit are Copycat and Tabletop [Mitchell, 1990][French, 1992]. The recognizer-evaluator architecture is based on their architectures. Both Copycat and Tabletop create micro-domain analogies in psychologically plausible ways. Both are based on models of emergent, high-level perception making use of a novel type of architecture that falls somewhere between the symbolic and connectionist paradigms. Top-level behavior emerges from many tiny stochastic computational actions that occur in parallel. Copycat and Tabletop model the way in which a small number of relevant concepts can be awakened from dormancy in order to understand and reason about a situation. Letter Spirit will make use of a similar architecture, but will focus on a larger-scale creative process that unfolds over an extended period of time.

Copycat and Tabletop have been exposited in a number of technical papers available from CRCC. Instead of discussing them here, we will focus on the way in which the Coderack/Workspace architecture will be applied to Letter Spirit.

The two main components of the architecture of the recognizer-evaluator are the Workspace and the Coderack. They can be roughly described as follows: The Workspace is the location of perceptual activity. It is the area where information from a variety of levels (ranging from quanta-level sensory data, through perceptual information about parts, labels, and roles, and eventually to the level of wholes) is built up and integrated. The Workspace provides the recognizer-evaluator with a short-term memory in which structures represent the state of the perceptual process. Our Workspace roughly corresponds to Hearsay II's Blackboard.

The Coderack is a "stochastic waiting room" where small computational agents wait before being allowed into the Workspace to do their work. It corresponds very roughly to a *queue* (in the operating-systems sense) where processes wait before being given their slice of CPU time. The difference between an OS queue and the Coderack is that agents are stochastically selected to run instead of having a deterministic order. The non-deterministic nature of the Coderack is a critical feature of our architecture — one of the features that gives rise to fluid behavior.

The Workspace — perceptual activity center of the recognizer-evaluator

The Workspace can be thought of as a busy construction site where structures of many sizes and shapes are built in multiple locations throughout the site. Independent work crews build the structures, occasionally removing obsolete ones in order to build (hopefully) better ones. The Workspace metaphor described in [Mitchell, 1990] is essentially the same as the cell cytoplasm metaphor described in [Hofstadter, 1983]. In the cytoplasm of a cell, enzymes carry out diverse tasks in multiple locations, building many hierarchically structured biomolecules, sometimes scavenging parts from other structures as needed.

At the beginning of a run, the Workspace consists only of raw data in the form

of quanta. Each item has no information connected to it. Over time, small localist perceptual agents called *glue-ants* crawl over the grid depositing "glue" on quanta intersection points. Glue is deposited stochastically, with angles getting less than straight intersections. When enough glue has been deposited, proto-parts are formed through the shaking process described above. The resultant parts are then labeled by many "labeling" codelets that attach syntactic labels to the parts. The codelets are very small probabilistic computational agents that act independently (codelets will be described more fully in the next section). Over time, items in the Workspace (e.g., parts and eventually roles) acquire various descriptive *labels* and together constitute *perceptual structures.*

When enough good structure has been built, the Workspace is invaded by codelets sent out by *roles.* Different roles spawn competing codelets to scout for role-instances on the grid. For a role to successfully match up with a part, it must both "fit" the chosen part in a reasonable way, and be consistent with other prior role/part couples in the Workspace. The Workspace develops nondeterministically, but not haphazardly. Structures must maintain coherence at a global level in order to survive. A useful image is that of a large structure (like a bridge) being built by a colony of hundreds of ants or termites. The ants work semi-independently, but cooperatively. Codelets (described below) correspond to the ants in the bridge metaphor. From the hundreds of tiny decisions that are made probabilistically, a coherent view of the data emerges.

The Coderack — stochastic control center of the recognizer-evaluator

In the Workspace, acts of gluing, labeling, scanning, matching, destroying, and so on are carried out by tiny computational agents called *codelets.* As many codelets run, their independent effects build upon one another into a collective behavior. The effect of each codelet considered by itself is insignificant. Codelets identify the features of structures in the Workspace. They also build and destroy such structures.

Codelets wait to run on the Coderack. Each codelet is assigned an urgency that is determined by estimating the importance of its possible action. Urgency values are based on how well a codelet's possible effect coheres with the structure built so far in the Workspace. Codelets that will most likely enhance the structures in the Workspace and dovetail with the emerging view will be assigned high urgency and have a good chance of getting to run. Codelets that do not seem particularly relevant will be assigned low urgency and will probably have to wait a long time to run.

Once some initial parts have been formed by gluing and shaking, an initial population of labeler codelets is placed on the Coderack. At each time step, one codelet is chosen to run from the Coderack. Codelet choice is probabilistic, and is based on the relative urgencies of all the codelets on the Coderack at the time of choice. The stochastic nature of codelet picking ensures that low-urgency codelets have at least some chance of running, while making sure that aimless processing is avoided.

Since codelets have very small effects, it is never critical that a particular codelet get selected. What does matter is that certain broad-stroked courses of action as a whole (called pressures) run faster than others. Probabilistic selection based on urgencies allows this to happen. This type of resource allocation in terms of processing power is an emergent phenomenon of Letter Spirit and is not determined in advance of a run.

During a run, new codelets are added to the Coderack. This ensures that the Coderack never drains completely away. New codelets can be posted as follow-ups to a codelet that has just completed its task, or may be posted as certain stages of processing are reached. Urgency is assigned to any new codelet as a function of its estimated promise. A follow-up codelet, for example, gets an urgency based on the success of the codelet that spawned it. The population of the Coderack is dynamically adjusted in response to the system's needs as judged by the amount and goodness of structure in the Workspace.

Though Letter Spirit will be implemented on a serial computer, and only one codelet will run at a time, the system is in principle a parallel system in which many different activities take place in parallel at different speeds. Viewing Letter Spirit as a parallel-processing system is justified since codelets work in localist and independent ways. The speed of particular abstract courses of processing in Letter Spirit is unpredictable in advance since urgencies drive the system. The speed at which any given course of action is explored is a statistical consequence of current processing.

5.3 Code

Work on implementation of the recognizer-evaluator has already begun and has progressed to the stage of part labeling. Code allowing the Coderack to select and run codelets has been completed. The program will be designed with a graphical interface so that activity in the Workspace is displayed on the screen as a run progresses. (Letter Spirit will be developed as an X program so that the code will be portable to multiple platforms.)

Other available tools being developed at CRCC for the Letter Spirit project include the Gridfont program and the 'a'-ness Probe. The Gridfont program is an X application that allows the user to graphically design, store, and manipulate gridfonts. Many of the 600 gridfonts we have designed to date have been entered into the Gridfont program's database. The 'a'-ness probe is a database of 'a's which are each given a rating according to their strength as 'a's. We have designed approximately 1500 'a's so far. The idea is to explore the space of gridfont 'a's in order to get a feeling for what being an 'a' is all about.

6 Letter Recognition in the Literature

There has been a large amount of work on letter recognition in both the fields of machine perception and cognitive psychology. Not surprisingly, the approaches have little common ground. Machine recognition stresses syntactic processing and prototype matching while cognitive psychology stresses the importance of high-level letter categories.

Letter recognition by machines can be classified into various divisions based on the type of recognition being done [Mantas, 1986]. The most relevant approach, in terms of the recognizer-evaluator, is known as Optical Character Recognition (OCR). Other approaches include on-line recognition (where timing information is available), and cursive-script and handwriting recognition where partitioning data into letters is a hard problem.

OCR involves parsing documents written in a variety of fonts into machine readable form. Although there are papers on OCR with amazing names like "On the Recognition of Printed Characters of *Any Font and Size*"[6] [emphasis added], and OCR machines are available commercially, a survey of the literature shows that the problem of OCR has not been satisfactorily solved. New methods are still being explored. A system that can recognize letters in many typefaces that it has never seen (or been trained on) has yet to be developed.

The OCR literature presents a variety of approaches ranging from using neural networks and classifier systems [Frey and Slate, 1990] [Guyon et al., 1989][Hinton et al., 1992][Lee and Oldham, 1990], through using the method of moments and other more standard machine perception tools [Cash and Hatamian, 1987][Mori et al., 1984] [Wang and Suen, 1984]. Approaches can be generally characterized as syntactic (in terms of the distinction developed above). Although error rates for many of the OCR systems above are low, they must be trained on any new font before recognition is possible. The recognizer-evaluator will differ substantially from standard machine letter-recognition approaches.

Cognitive psychologists studying letter perception emphasize the building up of higher-level structures from visual stimuli [Palmer, 1977][Palmer, 1978][Triesman and Gelade, 1980]. The cognitive psychology literature presents ideas that generally harmonize with the recognizer-evaluator model of letter perception. Cognitive psychologists recognize the critical influence categories and concepts have on letter perception. Their work explores what letter category boundaries are, how similarity of letters making up a letter category influences perception, and how letter categories interact with each other. Rob Goldstone's (as yet unpublished) work on letter perception[7] explores

[6] [Kahan et al., 1987] actually tested their model on six fairly standard fonts.

[7] I have learned about Goldstone's research through private correspondence.

just these issues, as does the work of Peter Podgorny [Podgorny and Garner, 1979]. Since psychological plausibility is an important aspect of the Letter Spirit project, we follow psychological research on letter perception with great interest.

7 Related Work

7.1 Typeface creation

7.1.1 DAFFODIL

[Nanard et al., 1986] have developed a typeface design program called DAFFODIL (Declarative Approach For FOnt Design by Incremental Learning). The DAFFODIL program is based on the unspoken assumption that style simply amounts to embellishment. The idea is simple — give the program 26 invariant letter "backbones", then allow the program to systematically replace the backbones with more complex patterns following style rules. Voilà — instant typeface creation! The resulting fonts are not bad either.

The problem with DAFFODIL as an AI program is that the computer is not doing anything creative at all because its actions are not *autonomous*. It is simply "doing what it was told to do". (Just exactly when a program is no longer "doing what it was told to do" is a tricky question. It may have something to do with nondeterministic architectures such as those of Copycat, Tabletop and Letter Spirit.) All creative aspects either are left up to the users of the program or are completely specified by the programmers. Like MetaFont, DAFFODIL is a sophisticated tool. People design the backbones and the ornate shapes to put on them. People specify the heuristics for associating a particular flowery shape with a section of the backbone. People look at the final product to make sure it is acceptable. The program barely does anything interesting at all. Its job amounts only to syntactic substitution and graphical display. The resulting font is not created by the *program*, it is created by a *person*, using the computer as a powerful tool.

DAFFODIL may well be a useful tool for some aspects of letter design, but we disagree with the philosophy of design that it espouses. DAFFODIL is based on the tacit assumption that style amounts to no more than simple ornamentation at the surface level. The message pervading the DAFFODIL approach is that a single alphabet template (i.e., set of geometrical backbones for all the letters), some nice flowery shapes, and a set of substitution rules are all there is to letter design. Very little is said about deep-level stylistic manipulation and transformation. It *is* possible since a DAFFODIL user can tamper with the backbones, but on its own, the program does no work at this level. Only people can manipulate the 26 letter backbones, and such tampering is not touted as one of the program's strengths, but is instead written

off as a necessary evil.

DAFFODIL's approach to letter design is diametrically opposed to ours. In Letter Spirit, the program, not a human, will be in control of the design process. The program will decide whether a letterform it is working on is acceptable as an 'a', and more specifically, whether the letterform is acceptable as an 'a' in the style currently under consideration. As mentioned above, there will not be 26 explicitly defined backbone shapes. Instead, Letter Spirit will work with fluid abstractions of letters that can respond to stylistic pressures, using them as inspiration for the actual letterforms that it creates.

7.1.2 The GridFont network

Grebert, Stork, Keesing and Mims approach letter design from a connectionist standpoint [Grebert et al., 1991a].[8] They even use our gridfont domain. Their attempt focuses on what they call "generalizing production from examples" using a simple feedforward network. Their model consists of a custom-designed three-layer network that is supposed to *learn* how to create gridfonts by being trained on a few examples.

The GridFont network discussed in [Grebert et al., 1991a] was trained on five gridfonts (*Standard Square, House, Slant, Benzene Right* and *Benzene Left*) until it was able to produce them all with only a small amount of error. Then it was supplied with and trained on fourteen letters from the gridfont *Hunt Four*. The network's task was to construct the twelve letters that it had never seen from the target gridfont.

Input to the network consists of a localist representation of both the letter and the font (this amounts to 32 units — 26 for the letter code and 6 for the font code). The hidden layer has 88 hidden units and is broken into two modules of 44 units each. One of the hidden-layer modules receives input from the letter nodes and the other from the font nodes. Activation from the "letter" hidden units feeds sideways into the "font" hidden units. Both sets of hidden units are specially connected to the output layer, which contains 56 units — one for each quantum. The output units are grouped in sections that correspond to the zones of the grid (see Figure 4 for a picture of the grid). Figure 10 shows the network architecture.

Connections between the hidden layer and the output layer are restricted for three reasons. (1) Restriction reduces the total number of connections, thereby decreasing simulation time and improving generalization. (2) Meaningless correlations between strokes in widely separated zones are prevented. Associations between quanta in the descender zone with quanta in the ascender zone are probably unnecessary and can be removed from the network. (3) Resources are more easily divided between zones such that greater computational resources (i.e., more weights) are given to the

[8] See also [Grebert et al., 1991b], [Grebert et al., 1991c], and [Grebert et al., 1992].

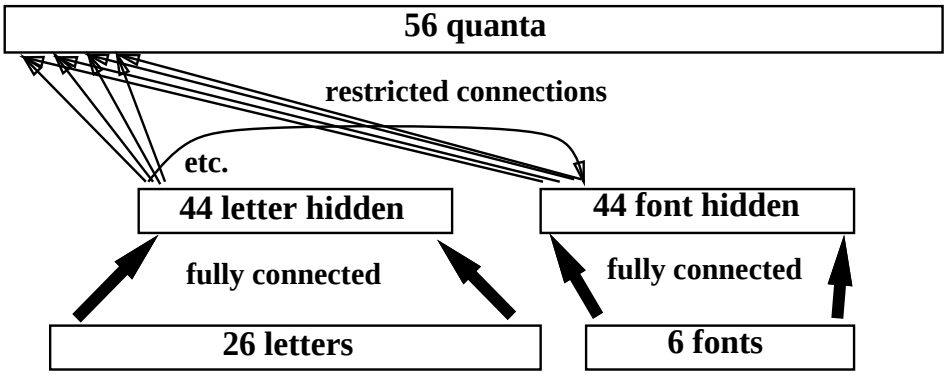

Figure 10: The GridFont network architecture. Each of the two input layer sections is fully connected only to its associated hidden layer. The projections from the hidden layers to the output layer are restricted.

central zone, which is the most important zone. Restricting the connectivity helps the network considerably in its creation task since a number of the possible mappings have been eliminated *a priori*.

The GridFont network's results are not terrible, especially considering that it has never been trained on the letters it is meant to create — but they are not good either. Figure 11 (from [Grebert et al., 1991a]) shows the target letters from *Hunt Four* and the actual output from the network. The network was trained on so many letters from its target gridfont (specifically, 'a', 'd', 'e', 'f', 'j', 'k', 'm', 'o', 'q', 'r', 's', 'u', and 'x') that most of the key stylistic features had already been designed by a person.[9] For example, the network had seen a 'd', so making a 'b' should have been easy. Even with all this information, it still came up with some rather bizarre looking letters, and one completely unrecognizable one.

One of the many weird things the GridFont network does is to put an extra quantum "jog" on its 'b'. This is something it is hard to imagine a human designer ever doing (especially in the context of *Hunt Four*). The network, however, lacking a realistic concept of 'b'-ness, can do nothing but consult the chart of associations it has coded into its weights. It turns out that most of the 'b's the network has seen so far happen to have the vertical quantum in the upper-right part of the central zone turned on. So it readily adds what is in this case turns out to be an "extra" quantum to its *Hunt Four* 'b'. Since the network is doing nothing more than pattern association, it has no conceptual foundation for design. It simply creates letters based on statistical similarities that it has "generalized out" of its training set. If concepts

[9] [Grebert et al., 1991c] selected their training set specifically in order to have this characteristic (see p. 387).

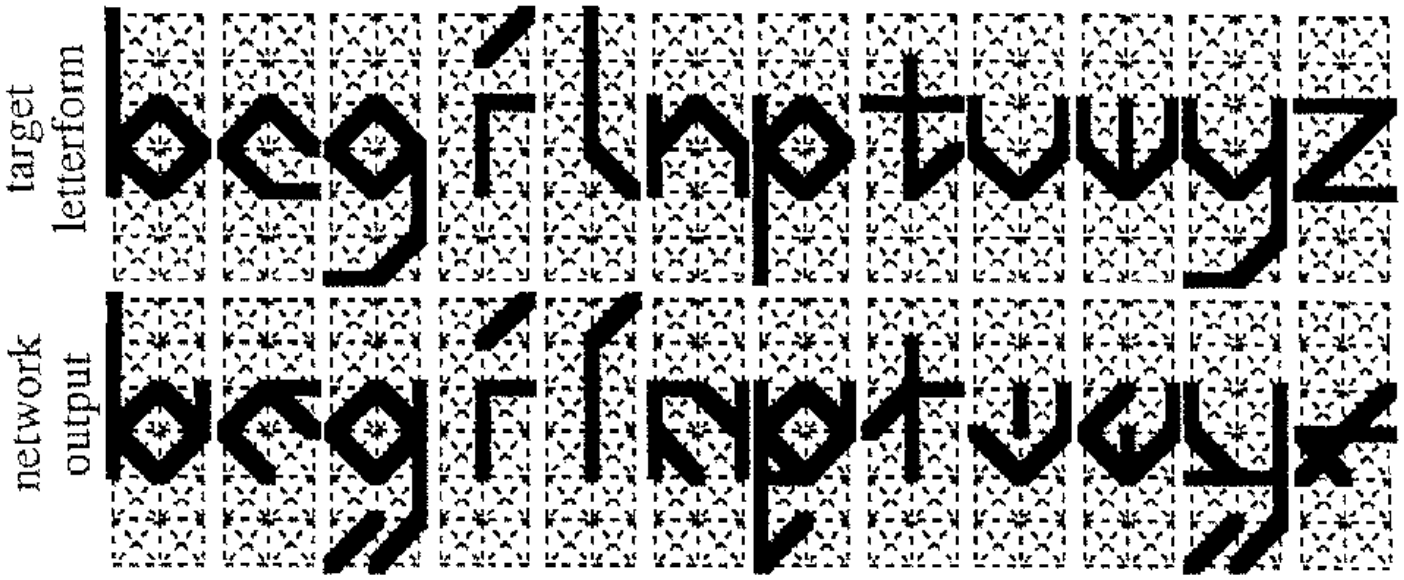

Figure 11: Target letterforms from *hunt four* (top row), and the output of the GridFont network (bottom row).

were merely statistical beasts, this might work, but this is not the case.

Since GridFont has no abstract *idea* about what it means to be a given letter, it does very naive things, including: sticking an extra quantum on the 'b','c','n', and 'y', making a very weird double closure on the 'p', using discontinuous descenders on the 'g' and 'y', breaking its 'v' into two parts, and creating a completely unrecognizable 'z'. There is obviously no idea behind the design of a letter here.

These results strengthen our claim that a conceptual basis is required for aesthetic gridfont design. Although the designers of GridFont might argue that their network has a conceptual basis (in the sense that it has a set of associations that roughly correspond to 'a'-ness), the problem is that the network has no *explicit* concepts which it can get to in order to manipulate them at a higher level. GridFont has no awareness of the internal structures and boundaries of its "concepts". In order to create 'a's and 'e's one needs to know *what it means* to be 'a's and 'e's, what 'a's and 'e's have in common with other letters, and what sets them apart. At the very least, some sort of perceptual filter should be applied after each candidate letter is created, so that too-bizarre letters get thrown away or get cleaned up. Such an approach would amount to connecting a fairly haphazard juggler-suggestor to a good recognizer-evaluator.

Although the GridFont network is not very good at gridfont design, its perfor-

mance might have been better had it been trained on more gridfonts. It might have been able to tune its statistical chart in such a way as to avoid really silly mistakes. Even so, we doubt that an architecture like the one used in the GridFont network could ever come close to the subtlety of human gridfont design. The approach is too simplistic.

The main problem with the GridFont approach is the complete lack of a temporal aspect. In Letter Spirit, any design decisions made during the creation of one letter will have a large amount of influence on the design all of the other letters in the gridfont (even the "finished" ones). The GridFont network has no such property. All of GridFont network's letters are created in parallel at exactly the same instant — they are all pushed out in a huge feed-forward gush.[10] GridFont's creators seem to have lost sight of the fact that temporality is an essential aspect of large-scale acts of creation like gridfont design.

All of this is not to say that every connectionist approach to letter design will necessarily suffer the same fate as the GridFont network. At the very least we can point out that at some level the human brain can be considered a connectionist architecture (albeit a powerful and sophisticated one). But the level of approach in GridFont is too statistical. We suspect that models making use of higher levels of abstraction (allowing for concept-level manipulation) and employing a temporally realistic approach would perform better.

7.2 Gridfont recognition

Since my thesis will deal almost exclusively with the recognition phase of the Letter Spirit project, it is important to build and test alternative architectures in order to assess the strengths and weaknesses of the one we have chosen to implement (of course, if there had been any models already developed for gridfont recognition I would have considered them too). We have spent a considerable amount of time researching straightforward connectionist and symbolic approaches to gridfont recognition. The hope is to create some basis for inter-architecture comparison and analysis. Both the NetRec and DumRec projects discussed below are on-going. More information about the two projects is available in [McGraw, 1990].

[10] In actuality, they are created one at a time since only one letter-node can be clamped during a cycle, but the design of one letterform has absolutely no affect on the design of the others.

7.2.1 NetRec — connectionism

We have completed several experiments in gridfont letter recognition using two- and three-layer feedforward connectionist networks.[11] The NetRec networks are trained (using backpropagation) on a variety of gridfonts and then tested on fonts they have not yet seen. Results show that connectionist networks can perform fairly well on recognition tasks. The ability to generalize about input patterns and create complex association maps between input patterns and their categories is especially useful in gridfont recognition. Instead of reporting specific results, we will simply discuss their broad implications for the Letter Spirit project.

Our networks use a localist representation for classification purposes. The input layer consists of 56 nodes, one for each quantum on the grid. If a quantum is "on" (i.e., it has a line drawn in its position), the quantum's associated node gets high activation. If it is "off", its node gets little or no activation. The output layer is simply a vector of 26 nodes, one for each letter of the alphabet. Hidden layers consist of anywhere from 0 to 120 nodes. The network is trained to associate patterns on its input layer with patterns on its output layer, using standard backpropagation of error.

Backpropagation is especially good at finding statistical regularities among what seem to be disparate category members. Thus if a network is trained on a large number of zany gridfont 'a's, it will discover many of the statistical similarities they share. It can use the statistical information to classify patterns it has not yet encountered.

A connectionist approach to letterform recognition would probably be better at recognizing letters without regard to their style than at recognizing letters while taking style into account. A network can classify its input pattern as a strong or weak member of a given letter-category by varying the degree of activation on the appropriate letter-node of its output layer. If a letterform is ambiguous, more than one node can be activated. Such a network could probably make good "objective" judgements regarding a mystery letter's category.

One problem with a connectionist approach is that although a connectionist network can classify letterforms as strong or weak, it cannot be judge them in terms of style. It is possible that the stylistic aspects of a proposed 'a' are completely wrong (with respect to other letters) even though the shape itself is a reasonable member of the category 'a'. The only way around this would be to train another network on style and somehow coordinate the two. But since an entire gridfont in the target style does not yet exist, and there is no way of talking about what style is in terms of some key concepts; not enough information exists for style training (as evidenced by the GridFont net). What is needed is a network that does analogy. Unfortunately,

[11] Dan Drasin, an undergraduate in Computer Science at IU, has been doing NetRec research under my direction for C390 credit.

analogy has yet to be successfully tackled in the connectionist paradigm, so die-hard connectionists will just have to wait.[12]

Another problem with NetRec is figuring out how to integrate a connectionist recognizer to the juggler-suggestor. Clearly a purely statistical approach (à la GridFont) will not work. Top-down semantic — or conceptual — pressure must be brought to bear at some stage of the juggling process.

7.2.2 DumRec — GOFAI

We have also written a messy-symbolic-AI gridfont-recognition program called *DumRec*. DumRec compares a mystery letter with a number of stored letters, counting agreements and disagreements of all sorts (which are computed with various levels of complexity), and making a decision based on a weighted comparison of microfeatures. DumRec's database of stored letters can be changed to include any number of different "training letters".

After reading in a data file containing several "training letters" and computing a property list for each letter, DumRec requests a mystery letter. When the mystery letter is presented, a property list is computed for it in exactly the same way. The mystery letter's property list is eventually compared, using a weighted metric, to the property lists of the training letters.

Dumrec has uses two types of comparison. The first is a direct approach in which the quanta of the mystery letter are directly compared to the quanta of each training letter. DumRec uses two comparisons of this type: a rating based on Hamming distance, and an "on"-bit rate which is the number of quanta shared between the mystery letter and a known letter. Points are awarded to each training letter based on how well it matches the mystery letter. A highly simplified version of DumRec called HamRec uses only the Hamming distance rating to guess the mystery letter's category. Its performance is significantly worse than DumRec's (fortunately).

All the other points awarded to training letters are based on comparison of the pre-computed feature lists of each training letter to the mystery letter's feature list. These features can be broken into three levels (based on the amount of abstraction involved in the computation).

Level-1 data involves a small amount of abstraction since level-1 properties are made up of multiple quanta. Level-1 data includes position information for a variety of simple angles and shapes. Numbers of certain types of shapes (like squares, triangles, lines, angles and closures) are also considered level-1 data.

Level-2 data consists of more abstract properties — namely, "mass", "gravity" and sectorization data. The grid has six sectors (the three zones divided into left and

[12]Doug Blank, a fellow cognitive science graduate student at IU, is rumored to be working on a connectionist analogy network.

right halves), each of which may or may not include quanta from the letterform. A letter's "gravity" is the number of the sector in which most of its quanta are found.

Level-3 data is the most abstract of all in the DumRec program. Level-3 properties include "point" and "tip" information. The top of the bar on the left side of the letter 'b' is an example of what we mean by a tip. "Points" are the 21 intersection points of quanta on the grid.

Once the property data have been calculated for the mystery letter, the matching process begins. DumRec calculates a score for each training letter based on its relation to the mystery letter. The score is a weighted ranking of the match of the property lists. Needless to say, the weights play a crucial role in DumRec's performance. Modifying the weights is easy and is an important part of "tuning" DumRec.

DumRec has a graphical interface which makes understanding its results, entering mystery letters, and seeing what a data set looks like fairly straightforward. We are still playing with DumRec in order to discover its personality. Interestingly, it is fairly good at gridfont letter recognition, most of the time either guessing the correct letter or picking a "reasonable" wrong one. Comparing DumRec's performance to the performance of the recognizer-evaluator module of Letter Spirit will be both informative and fun.

8 Conclusion

Letter Spirit is meant to model the central mechanisms of creativity. We feel that if AI and cognitive science are to understand the workings of the human mind (or of any mind at all), they must focus more attention on concepts. Letter Spirit addresses the relation of concepts to creativity. It explores new ground in both AI and cognitive science. A complete implementation of Letter Spirit is interesting both as a way of imparting creativity to a machine, and as a plausible model of human creativity.

Implementation of the recognizer-evaluator is an important milestone on the road towards Letter Spirit. Letter Spirit is such a large project that the only way to expect it to be implemented is to break it into manageable pieces, each of which can stand alone as a thesis project. Future work at CRCC will include implementation of the juggler-suggestor module and the eventual integration of the juggler-suggestor and the recognizer-evaluator into a full-scale Letter Spirit program. I will attempt to implement a preliminary version of the juggler-suggestor module in conjunction with the recognizer-evaluator, but it will most likely be primitive.

The recognizer-evaluator will serve as a test of the applicability of the Coderack/Workspace architecture (shown to work in both Copycat and Tabletop) to the perceptual end of large-scale creative tasks. The parallel stochastic processing mechanisms of Letter Spirit fall under the rubric of *emergent computation*, in which top-level

behavior emerges as a statistical consequence of many small computational actions. Like Copycat and Tabletop, Letter Spirit will occupy a level of cognitive modeling somewhere between connectionism and symbolic AI — the level at which we feel the most useful work on understanding perception and the fluidity of concepts can be done.

The recognizer-evaluator offers a new model of letter perception based on the idea of letterforms as conceptual entities. The recognizer-evaluator will break new ground in letter recognition and may prove useful as a model of machine perception even when considered on its own, apart from the Letter Spirit program. The recognizer-evaluator will provide a model of high-level perception and its emergence from low-level independent micro-actions. It is our contention that high-level perception of this emergent character is required for success in letter recognition.

Acknowledgments

Special thanks to my adviser Doug Hofstadter. This paper relies heavily on his ideas about cognitive science and on his meticulous proofreading. Also big thanks to my friend Jim Marshall who suffered through several drafts at FARGlets. May he recover from his nightmares of somnambulatory defenestration. The rest of FARG was most useful too, especially Bob French.

References

[Altshuller, 1988] Altshuller, G. (1988). *Creativity as an Exact Science: The theory of the solution of inventive problems.* Studies in Cybernetics. Gordon and Breach Science Publishers, New York.

[Bongard, 1970] Bongard, M. (1970). *Pattern Recognition.* Hayden Book Co, Spartan Books, Rochelle Park, NJ.

[Cash and Hatamian, 1987] Cash, G. and Hatamian, M. (1987). Optical character recognition by the method of moments. Computer Vision, Graphics, and Image Processing, 39:291–310.

[Chalmers et al., 1990] Chalmers, D., French, R., and Hofstadter, D. (1990). High-level perception, representation, and analogy: A critique of artificial intelligence methodology. Technical Report 49, Center for Research on Concepts and Cognition, Indiana University, Bloomington, IN. Also to appear in *Journal for Experimental and Theoretical Artificial Intelligence.*

[Chamberlain, 1984] Chamberlain, W. (1984). *The Policeman's Beard is Half-Constructed: computer prose and poetry.* Warner Software/Books, New York.

[Crozier and Chapman, 1984] Crozier, W. and Chapman, A., editors (1984). *Cognitive Processes in the Perception of Art.* Advances in Psychology (19). North Holland, Amsterdam.

[Erman et al., 1980] 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, 1992] French, R. (1992). *Tabletop: An emergent stochastic computer model of analogy-making.* PhD thesis, University of Michigan, Ann Arbor, Michigan.

[Frey and Slate, 1990] Frey, P. and Slate, D. (1990). Letter recognition using holland-style adaptive classifiers. Unpublished report.

[Grebert et al., 1991a] Grebert, I., Stork, D., Keesing, R., and Mims, S. (1991a). Connectionist generalization for production: An example from gridfont. In Proceedings of the 1991 International Joint Conference on Neural Networks (IJCNN).

[Grebert et al., 1991b] Grebert, I., Stork, D., Keesing, R., and Mims, S. (1991b). Connectionist generalization for production: An example from gridfont. Technical Report CRC-TR-91-20, Ricoh California Research Center, Menlo Park, CA.

[Grebert et al., 1991c] Grebert, I., Stork, D., Keesing, R., and Mims, S. (1991c). Network generalization for production: Learning and producing styled letterforms. In Proceedings of the Neural Information Processing Systems Conference (NIPS) 1991.

[Grebert et al., 1992] Grebert, I., Stork, D., Keesing, R., and Mims, S. (1992). Connectionist generalization for production: An example from gridfont. Neural Networks, 5.

[Guyon et al., 1989] Guyon, I., Poujaud, I., Personnaz, L., and Dreyfus, G. (1989). Comparing different neural network architectures for classifying handwritten digits. In Proceedings of the International Joint Conference on Neural Networks (IJCNN-89), pages II, 127–132, Washington, D.C.

[Hammond, 1989] Hammond, K. (1989). *Inside Case-Based Reasoning*, chapter 6: CHEF, pages 165–211. Lawrence Erlbaum and Associates, Hillsdale, New Jersey.

[Haugeland, 1987] Haugeland, J., editor (1987). *Mind Design: Philosophy, psychology, artificial intelligence.* MIT Press, Cambridge, Mass.

[Hinton et al., 1992] Hinton, G., Williams, K., and Revow, M. (1992). Adaptive elastic models for hand-printed character recognition. Presented at Cogsci 1991, and posted to Neuroprose.

[Hofstadter, 1979] Hofstadter, D. (1979). *Gödel, Escher, Bach: an Eternal Golden Braid.* Basic Books, New York.

[Hofstadter, 1983] Hofstadter, D. (1983). The architecture of jumbo. In Proceedings of the International Machine Learning Workshop, Monticello, IL.

[Hofstadter, 1985a] Hofstadter, D. (1985a). *Metamagical Themas*, chapter 10: Parquet Deformations: A Subtle, Intricate Art Form. Basic Books, New York.

[Hofstadter, 1985b] Hofstadter, D. (1985b). *Metamagical Themas*, chapter 26: Waking Up from the Boolean Dream: Subcognition as computation, pages 631–665. Basic Books, New York.

[Hofstadter, 1985c] Hofstadter, D. (1985c). *Metamagical Themas*, chapter 23: On the seeming paradox of mechanizing creativity, pages 547–603. Basic Books, New York.

[Hofstadter, 1985d] Hofstadter, D. (1985d). *Metamagical Themas*, chapter 12: Variations on a theme as the crux of creativity, pages 232–259. Basic Books, New York.

[Hofstadter, 1987] Hofstadter, D. (1987). Fluid analogies and human creativity. Technical Report 16, Center for Research on Concepts and Cognition, Indiana University, Bloomington, IN.

[Hofstadter and Mitchell, 1988] Hofstadter, D. and Mitchell, M. (1988). Concepts, analogies, and creativity. In Goebel, R., editor, CSCSI-88: Proceedings of the Seventh Biennial Conference of the Canadian Society for Computational Studies of Intelligence, pages 94–101. Edmonton, Alberta, Canada.

[Hofstadter and Mitchell, 1991] Hofstadter, D. and Mitchell, M. (1991). The copycat project: A model of mental fluidity and analogy-making. Technical Report 58, Center for Research on Concepts and Cognition, Indiana University, Bloomington, IN. To appear in Barnden, J. and Holyoak, K. *Advances in Connectionist and Neural Computation Theory. Volume 2: Analogical Connections.*

[Holland et al., 1986] Holland, J., Holyoak, K., Nisbett, R., and Thagard, P. (1986). *Induction.* MIT Press, Cambridge, Mass.

[Kahan et al., 1987] Kahan, S., Pavlidis, T., and Baird, H. (1987). On the recognition of printed characters of any font and size. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-9(2):274–288.

[Kahneman and Miller, 1986] Kahneman, D. and Miller, D. (1986). Norm theory: comparing reality to its alternatives. Psychological Review, 93:136–153.

[Knuth, 1982] Knuth, D. (1982). The concept of a meta-font. Visible Language, XVI(1):3–27.

[Kolodner and Penberthy, 1990] Kolodner and Penberthy (1990). A case-based approach to creativity in problem-solving. In Proceedings of the Twelfth Annual Conference of the Cognitive Science Society, pages 978–985. Lawrence Earlbaum Associates.

[Langley et al., 1987] Langley, P., Simon, H., Bradshaw, G., and Zytkow, J. (1987). *Scientific discovery: Computational explorations of the creative process.* MIT Press, Cambridge, Mass.

[Lee and Oldham, 1990] Lee, M. and Oldham, W. (1990). Font recognition by a neural network. International Journal of Man-Machine Studies, 33:41–61.

[Lenat, 1982] Lenat, D. (1982). *Knowledge-Based Systems in Artificial Intelligence*, chapter AM: Discovery in mathematics as heuristic search, pages 1–225. McGraw-Hill, New York.

[Lenat, 1983] Lenat, D. (1983). Eurisko: A program that learns new heuristics and domain concepts. Artificial Intelligence, 21(1,2):61–98.

[Mantas, 1986] Mantas, J. (1986). An overview of character recognition methodologies. Pattern Recognition, 19(6):425–430.

[Mantas, 1987] Mantas, J. (1987). Methodologies in pattern recognition and image analysis — a brief survey. Pattern Recognition, 20(1):1–6.

[McClelland and Rumelhart, 1988] McClelland, J. and Rumelhart, D. (1988). *Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence*, chapter An interactive activation model of context effects in letter perception: part 1, an account of basic findings, pages 580–596. Morgan Kaufmann, San Mateo, CA.

[McCorduck, 1991] McCorduck, P. (1991). *Aaron's Code: meta-art, artificial intellgence and the work of Harold Cohen.* Freeman, New York.

[McGraw, 1990] McGraw, G. (1990). Dumrec lives — birth of a project with multiple programalities. Center for Research on Concepts and Cognition, Indiana University, internal document.

[Mitchell, 1990] Mitchell, M. (1990). *Copycat: A computer model of high-level perception and conceptual slippage in analogy making.* PhD thesis, University of Michigan, Ann Arbor, Michigan.

[Mori et al., 1984] Mori, S., Yamamoto, K., and Yasuda, M. (1984). Research on machine recognition of handprinted characters. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6(4):386–405.

[Nanard et al., 1986] Nanard, M., Nanard, J., Gandara, M., and Porte, N. (1986). Declarative approach for font design by incremental learning. Technical report, Centre de Recherche en Informatique de Montpellier, Montpellier, France.

[Palmer, 1977] Palmer, S. (1977). Hierarchical structure in perceptual representation. Cognitive Psychology, 9:441–474.

[Palmer, 1978] Palmer, S. (1978). Structural aspects of visual similarity. Memory & Cognition, 6(2):91–97.

[Podgorny and Garner, 1979] Podgorny, P. and Garner, W. (1979). Reaction time as a measure of inter- and intra-object visual similarity: Letters of the alphabet. Perception & Psychophysics, 26(1):37–52.

[Rieger, 1975] Rieger, C. (1975). *Conceptual Information Processing*, chapter Conceptual memory and inference. North-Holland, Amsterdam.

[Schank, 1986] Schank, R. (1986). *Explanation Patterns: Understanding Mechanically and Creatively.* Lawrence Erlbaum Associates, Hillsdale, New Jersey.

[Schank and Childers, 1988] Schank, R. and Childers, P. (1988). *The Creative Attitude: Learning to ask and answer the right questions.* MacMillan, New York.

[Schank and Leake, 1989] Schank, R. and Leake, D. (1989). Creativity and learning in a case-based explainer. Artificial Intelligence, 40(1-3):353–385. Also in Carbonelle, J. (ed.), *Machine Learning: Paradigms and Methods*, MIT Press, Cambridge, MA, 1990.

[Triesman and Gelade, 1980] Triesman, A. and Gelade, G. (1980). A feature-integration theory of attention. Cognitive Psychology, 12(12):97–136.

[Tversky, 1977] Tversky, A. (1977). Features of similarity. Psychological Review, 84:372–352.

[Vickery, 1983] Vickery, R. (1983). *Sharing Architecture.* University of Virginia Press, Charlottesville, VA.

[Wang and Suen, 1984] Wang, Q. and Suen, C. (1984). Analysis and design of a decision tree based on entropy reduction and its applicatoin to large character set recognition. IEEE Transcations on Pattern Analysis and Machine Intelligence, PAMI-6(4):406–417.