Letter Spirit (part two): Modeling Creativity in a Visual Domain

John A. Rehling

Submitted to the faculty of the Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the Department of Computer Science
and the Cognitive Science Program
Indiana University

July 2001

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Dr. Douglas R. Hofstadter
(Principal Adviser)

Dr. Michael Gasser

Dr. Robert Goldstone

Dr. Robert F. Port

Bloomington, Indiana
May 2000.

ABSTRACT

This thesis presents work on the Letter Spirit project. The Letter Spirit project aims to model central aspects of human high-level perception and creativity on a computer, focusing on the creative act of artistic letter-design. The aim is to model the process of rendering the 26 lowercase letters of the Roman alphabet in many different, internally coherent styles. Two orthogonal aspects of letterforms are basic to the project: the *categorical sameness* possessed by instances of a single *letter* in various styles (*e.g.*, the letter 'a' in Times, Palatino, and Helvetica), and the *stylistic sameness* possessed by instances of various letters in a single style, or *spirit* (*e.g.*, the letters 'a', 'k', and 'q' in Times alone). Starting with one or more seed letters representing the beginnings of a style, the Letter Spirit program attempts to create the rest of the alphabet in such a way that all 26 letters share the same style. Letters in the domain are formed exclusively from straight segments defined by a sparse grid in order to make decisions smaller in number and more discrete. This restriction allows much of low-level vision to be bypassed and forces concentration on higher-level cognitive processing, particularly the abstract and context-dependent nature of letter concepts.

ACKNOWLEDGMENTS

In a sense, this work began somewhere out near Jupiter.

In the summer of 1979, just before Doug Hofstadter was to begin considering the idea of a computer program that designed typefaces, I was at the public library in my hometown of Mount Vernon, Ohio. My parents took me there frequently, and I was with them there that night, a few weeks after I had finished fourth grade.

As we were leaving, I saw a magazine with a remarkable image on the cover — five worlds. One was familiar to me as Jupiter. The other four were utterly strange to me. Four months earlier, anyone else would have had to have said the same. These were the Galilean satellites of Jupiter, known for 370 years as points of light or little more; then the Voyager 1 spacecraft gave them faces. The cover of the June 1, 1979 issue of *Science* showed those faces to me, and blew bellows on the fire of my curiosity.

Both causally and metaphorically, this brought me to Letter Spirit, by a long and winding road. I talked my parents into paying for subscriptions to countless science magazines, though the articles often (if not usually) went over my head at that age. For indulging my love of books and learning, at great expense to them (far more than just a few subscriptions, once tuition kicked in), I thank my parents.

A little over a year later — it would have been near the end of 1980 — an issue of *Scientific American* appeared in our mailbox and introduced me to Douglas R. Hofstadter (I may be the only FARGonaut who didn't read *GEB* first). I have distinct memories of my explorations of self-referential sentences as I roved the streets of Mount Vernon. And I remember once reading Doug's column on a sofa — not only

the sofa nearest the door next to our mailbox, but the *end* of that sofa nearest the door, because I wanted to sit and start reading as soon as possible. This, I believe, attests as much to the appeal Doug's column had for me as to my sloth.

I could expound at length on where things went from there. I recall my friend Ben Kottler and me reading *GEB* during our senior year of high school. I remember walking down the Reserve Corridor of Dartmouth's library, musing over how to render fonts in coarser and coarser grids when I should have been doing homework. And from the time I got to Indiana on, innumerable times that Doug Hofstadter buoyed my spirits and my mental faculties. I picked a college major when I was eighteen based on the influence Doug's writings had on me, and the direction of my life has been set by that. I've been very lucky to get to work with him.

The project I did my dissertation work on was, in a previous life, Gary McGraw's project. Gary was an elder FARGonaut when I arrived at IU, and during the three years we overlapped, we had some good times in the lab, around a bonfire or two, and as far away as a couple of daytrips in Italy. I could probably write a whole thesis about the tremendous boon that the foundation he created for Letter Spirit was for me in my work. His thesis — I kept one copy at home, and one at the office — was Biblelike for me at key points in my graduate career. Stealing his techniques and templates in LaTeX, idraw, ghostview and other UNIX programs saved me tons of time. And his impressive record as a published grad student was a great example to try to follow.

During the time that most of my work on Letter Spirit was done, the majority of days went about the same way. I arose, walked from Second Street across Indiana's campus with my dog, Rikki, and headed over to CRCC. If this happened to be morning or afternoon, Helga Keller was probably in, and a fair percentage of the time, we compared notes on the news of the lab, the town, or the world, and then I headed upstairs. Helga also made everything that needed to work (except the computers in the center) work. Insurance plans, travel reimbursement, lab politics, probably the Sun and the Moon — Helga made things go.

And if you have business with Faber — or Faber —
I'll give you this tip, and it's worth a lot more
You'll save yourself time, and you'll spare yourself labour
If just you make friends with the Cat at the door.
— T. S. Eliot

Helga is the Cat at the door. That's the first thing to know about CRCC. I'll miss her personally and as an administrative juggernaut.

Often, I must say, I began my day at night, and then there was usually just an empty building waiting for me at the office. But whether I was on my walk north to work, at work, or on my walk south back home, the greatest invariant, by far, was Rikki, who had no idea what I was doing while sitting at my desk, and had no idea why we went, day after day. She always was eager to go, though, because it meant a mile of walk through prime rabbit territory. And just as eager to leave. There was a nice duality, in a way, in that the walks to and fro let Rikki search for what she was looking for (I think she may have bagged five or six kills over about as many years, but I may be undercounting). And sometimes, this delayed me, and I had to wait patiently while she scouted a shrub, or ran amok through Dunn Woods. At the office, she waited patiently while I did my thing. If Rikki had had her way, I'm sure the walks would have lasted hours, and the stopover at the office only minutes, instead of the other way around. I got the better of you, Rik, and I know it.

At opposite ends of my grad career were two generations of FARGonaut, and I a middle generation unto myself. Jim Marshall and I probably had the greatest overlap, and we worked and did the opposite for many hours — at CRCC, in numerous places around Italy, and most recently, when he came back to town for his successful thesis defense. Jim was a part of the old guard, and they — Dave Chalmers, Pei Wang, Steve Larson, Morten Christiansen, Jim and Gary — were great role models, and in many feverish discussions during the 1992-1993 year, were a big part of my grad upbringing.

The younger (not always literally) generation of FARGonauts also did a lot for my grad career. Hamid Ekbia, Harry Foundalis, and Jun Luo were the audience for most of my Letter Spirit talks, both on the group level and sometimes one-on-one. If you want to know anything well, teach it. The insightful questions from these guys were often the guarantor that I knew what I was talking about.

And I have to thank IU's crackerjack faculty in cognitive science. Mike Gasser, David Leake, and Greg Rawlins all taught courses that were not only on-target for my interests, but were so well-run that I would have taken them if the subject had been how paint dries. Rob Goldstone was a professor, committee member, colleague, mentor, and friend, and had an incredibly broad role in my Bloomington life. I also had some great courses with John Kruschke, Alice ter Meulen, and Joel Steinmetz, and the committee support of Bob Port.

Outside the work realm, many people (by phone, by e-mail, or live and in person) gave me support during my work. A comprehensive list would be silly to attempt. J. P. Deuble, Doug Eck, Joe Johnston, Pat Kalaher, Adrienne Anderson, Tony and Andrea Chemero, John Steinhauer, Alan Keahey, John Guzowski, Daxman Collins, Steve and Anna Banbury, Steve and Jen Rose — I'll stop there with no slights intended. If a few of you hadn't had such damn long names, I could have fit more in. A group mention goes to the Blooming Fools Hash House Harriers, whose brand of insanity was oh-so-right for me — running and beer.

God has had some kind of role in things here, if for no other reason than that I saw the role it had for my late grandmother Margaret Baker Ressler, the best person I'll ever know.

I said that the pictures of the moons of Jupiter had a metaphorical connection to Letter Spirit. It's like this: There were some featureless infinitesimal points, and a voyager swept by them and turned them into colorful diverse worlds. I hope that Letter Spirit is a voyager that gives us some best-yet pictures of the globes on our shoulders.

John Rehling
Bloomington
April 2000

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Part I

Foundations

CHAPTER ONE

Introduction: Letter and Spirit

1.1 Introduction

The Letter Spirit project aims to investigate human creativity by means of a computer model that operates in a rich microdomain. The comparison of the model's behavior with that of people creating in the same domain allows an assessment of the strengths and shortcomings of the approach. The design of *gridfonts* — skeletal typefaces rendered on a grid of 56 segments per letter — is the point of comparison between the creativity of humans and that of the Letter Spirit program.

The project was conceived of by Douglas Hofstadter in 1979, who designed the grid, the first gridfonts, and the broad outlines of the architecture in the early 1980s [Hofstadter 1985].[1] Only many years later did Letter Spirit become an active area of research in the Fluid Analogies Research Group ("FARG"). Programming was begun by Gary McGraw in the early 1990s, culminating in his dissertation work on a letter-recognizing program called the Examiner [McGraw 1995]. Since then, programming

[1]There is a rich tradition of grids as an organizing element in design and vision science alike; a partial history appears in Vitz and Glimcher [1984].

efforts by myself, in consultation with Douglas Hofstadter, have led to a program fulfilling many — but far from all — of the original goals [Hofstadter and FARG 1995].

1.2 Letter: The alphabet

A-B-C... Fallin' in love with you was easy for me
And you can do it, too; it's easy (it's so easy)
Like takin' candy from a baby...
— 1-2-3
Words and music by Len Madara, David White, Leonard Borisoff, Brian Holland, Lamont Dozier, and Eddie Holland

Letters constitute a complex phenomenon, but the mastery that literate people have of it means that it is easily taken for granted, and its complexity overlooked. The brief overview here makes explicit a number of issues regarding letters.

The Roman alphabet that English uses is the product of a long and frequently branching evolution in written communication. It has clear ancestors such as Phoenician, siblings in other versions of the Roman alphabet (as used for languages such as German, Vietnamese, and hundreds of others), cousins in Cyrillic and Hebrew (and many other alphabets), and possible cousins in Devanagari and other scripts of South Asia. Some other systems of writing, such as Maya and Chinese, appear to have evolved completely independently. And some writing systems, such as the Cherokee syllabary, have been deliberately designed using the Roman alphabet as a partial inspiration [Coe 1992].

The line of ancestry for English's Roman alphabet may have begun with graphical representations of events, gradually becoming a pictographic writing system (by 2000 B.C.), and eventually a syllabary, in which picture-signs that once represented whole words became signs for syllables. At this point, writing assumed a level of abstraction long since developed in spoken language: an arbitrary, culturally transmitted [Hockett 1960] set of relationships between signs and meanings. While painting is considered to have first attempted nonrepresentational works — those in which graphical images are not intended to have real-world referents — with Kandinsky in the twentieth century, letterforms have explored the realm of pure abstraction for millennia (as have other flourishes and patterns used in design, both within the Western tradition and elsewhere).

In time, some early ancestor of our alphabet became probably the first writing system that had individual symbols on the level of phonemes, rather than syllables. This allowed humans to write with a script that had only twenty to forty signs, rather than the many dozens required by syllabaries. This occurred by 1500 B.C. with the Phoenician script, which is the ancestor of the Roman alphabet (although the Phoenician language is not a genetic ancestor of Latin or English). The alphabet was then carried over by the borrowing of letters from Phoenician to Greek (by 700 B.C.), with the new Greek letters based on Phoenician letters corresponding roughly to the same sounds. This was possible only for consonants, as Phoenician did not record vowels, for reasons peculiar to the Semitic languages. The Greek alphabet acquired its vowels from some letters used in Phoenician for consonants. Later borrowing (by 250 B.C.) brought the alphabet to Latin, including Roman Britain, and this alphabet evolved a bit as it was eventually applied to English [Meggs 1998]. Even hundreds of years ago, something very similar to our current alphabet was in place, though a few changes have occurred more recently. Today, the Roman alphabet, in all its many versions, is the standard for a large number of languages in all parts of the world. Of the world's major languages, only English and Dutch use this exact set of 26 letters without any common additional marks, such as accents or umlauts (except in rare cases).

English has — like most languages, though to an unusual extent — drifted from the original conception of the alphabet as a one-to-one sign-to-phoneme set representing the sounds of a language. Still, letters basically exist at the level of the phoneme (there are irregularities; sometimes a short sequence of letters maps to a short sequence of phonemes). Because phonemes do not, usually, in isolation, have a particular meaning, letters are very abstract things indeed.

It was not always so. The Roman letters can be traced to their pictorial roots (via the ancestral line through Greek, Egyptian and Phoenician), and the original pictographic intention of each Roman letter is given in Table 1.1. Lowercase forms evolved later, based (sometimes loosely) on their uppercase counterparts.

The important point that will bear on future sections of this thesis is that letters do not resemble (rather, in the overwhelming majority of situations, are not intended to resemble, and are not expected to resemble) real-world objects.[2] This distinguishes alphanumeric characters from other forms of graphical representation. Thus, except in unusual cases, creation of letters is governed only by a pressure to make the letter recognizable and a pressure to make the letter match a particular style. The FARG account [Hofstadter 1985; McGraw 1995] of letterforms and letter categories appears in the next section.

1.3 Letter: Roles

Whether one wishes to consider the alphabet then or the alphabet now, letters are not, and have never been, defined as exact graphical shapes. Rather, as human artifacts drawn by often-unsteady hands using imperfect media, letters, both in their

[2]For the historical alphabetist eager to find a trace of the pictographic in modern English, the only place to turn is the word "door", better visible when written "Door". The almost-rectangular "D" was in fact designed to represent a door, and the name–sign relationship essentially survived the tumultuous and lengthy history of English and the Roman alphabet (one may say it died and was reincarnated).

A	ox (head)
B	house
C	boomerang
D	door
E	shouting (for joy)
F	hook
G	from C
H	rope (twisted)
I	hand
J	from I
K	open hand
L	staff
M	water (waves)
N	snake
O	eye
P	mouth
Q	monkey?
R	head
S	tooth
T	checkmark
U	support pole
V	from U
W	hook
X	fish
Y	from U
Z	arrow

Table 1.1: Each letter in our alphabet began as the graphical representation of an object.

rendering and their recognition, have a measure of variability. When one learns the alphabet, one learns abstract concepts of how letters may appear; actual letters on stone tablets, pages, or computer screens are exact shapes that merely instantiate a particular category in a particular manner. There is often deliberate play and pleasure involved in the casual writing or deliberate design of letters, and the sense of playfulness has arguably grown over the centuries. The potential variety in letterforms shows how flexible letter categories are for people, and how unrelated to any specific shapes.

Work on the Letter Spirit project has been guided by the belief that letters are

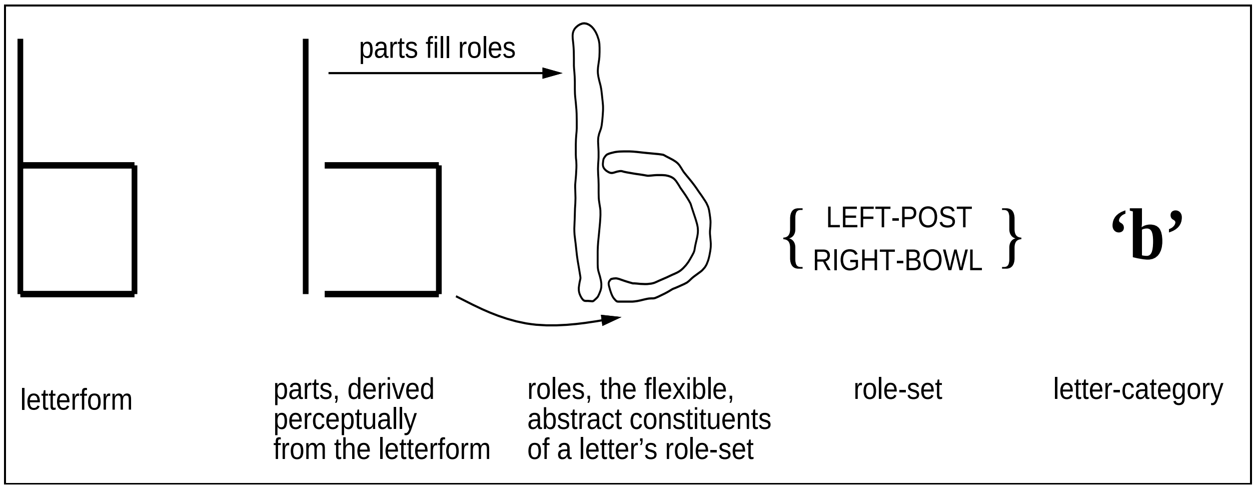
parts fill roles
LEFT-POST
RIGHT-BOWL
'b'
letterform
parts, derived perceptually from the letterform
roles, the flexible, abstract constituents of a letter's role-set
role-set
letter-category

Figure 1.1: Conscious perception of letters involves a hierarchy built around roles.

represented by *roles* — abstract concepts on a level beneath that of letters. Letters are defined by one or more *role-sets*, which specify which roles make up the letter and how they ought to touch or otherwise relate spatially to one another. An actual physical rendering of a letter — a *letterform* — can be decomposed into physical *role-fillers*, which satisfy (to some degree) the definitions of their corresponding roles. It is important to note that role-fillers are exact shapes, but roles are not. Roles are concepts, flexible enough that a great variety of shapes may serve as role-fillers for them. The role hypothesis and support for it are explained in greater detail in McGraw [1995]. It is summarized briefly in Figure 1.1; subsequent chapters offer greater elaboration on the theory and how it is implemented in Letter Spirit. In this thesis, it is argued that the perception of letters is based upon their decomposition into roles only in the case of *conscious* perception of letters, and that other mechanisms are at work in unconscious perception. The two putative kinds of letter perception are the subject of Chapter 2.

1.4 The domain

The Letter Spirit project aims to use typeface design as the vehicle to explore creativity, the more general object of primary interest. But even typeface design, as much as it is a relatively controlled and regulated domain, involves a daunting amount of complexity and of niggling, potentially microscopic detail. In principle, there is no limit to the fineness of a font. In order to put the emphasis clearly on the conceptual level, a level that appeals as much to the cerebrum as to the retina, the project concerns itself not with fonts, but gridfonts.

Gridfont letters (henceforth: "gridletters") are rendered on a 3×7 grid that consists of the 56 possible horizontal, vertical, and diagonal line segments (henceforth: "quanta") connecting adjacent points, as seen in Figure 1.2. (This figure appears on page 25 and will be a useful reference while reading other parts of the thesis, as well.) Many terms from conventional type design [Jaspert, et al. 1986] can be expected to apply; other terms specific to the project are added where convenient. Thus, the terms "x-height", "baseline", "ascender", and "descender" are borrowed from their common usage in typeface design, while other latitudes and longitudes are also given specific names, something that makes sense only with a discrete grid.

Two additional concepts that are especially useful in the grid domain are that of the "o-ring" and "post-and-bowl letters". The o-ring is simply the closed figure that, in many gridfonts, makes up the gridfont's exact 'o', and that often occurs within the 'b', 'd', 'g', 'p', and 'q'. Post-and-bowl letters, meanwhile, are exactly those just mentioned (not counting the 'o', which is all bowl and no post). The grid's coarse grain leads to alphabets in which a shape is repeated exactly in multiple letters, and some gridfonts have a large part of their identity tied up in the recurring use of a certain o-ring.

It has already been argued [Hofstadter and FARG 1995] that the gridfont domain is still sufficiently rich for the study of human creativity and this argument will be made more forcefully in this document. By the end, it should be evident that, for all that can be said about gridfonts constituting a microdomain, it is very hard to do justice to the domain within the scope of two (or a few) doctoral dissertations. It is hoped that this work generates a significantly detailed and accurate model of creativity; but it is no less important that its deficiencies be laid bare, and that some light be cast on what further roads must be followed to advance understanding. This thesis's discussion of a program that has consumed eight programmer-years of effort and nevertheless still leaves so much uncaptured should promote the conclusion that this is a very complex domain, and if it is considered “micro”, then the domains that one can be confident of modeling completely must be “nano” or “atto”.

The immense variety available in the gridfont domain is best made clear by illustration, and thus, three pages displaying 23 complete gridfonts *designed by humans* appear at the end of this chapter as Figure 1.3, Figure 1.4, and Figure 1.5. This does more than show impressive variety, however. It also introduces some of the leading “characters” in this drama, for these 23 gridfonts (and subsets thereof) make up the test set of choice for the Letter Spirit program throughout the thesis. Some are very difficult, and elude the program's ability to perceive or mimic. On most, the program enjoys at least partial success. The test set, like all manners of evaluation in this thesis, has been chosen to show off the program's strengths and weaknesses with the greatest clarity possible. The set should also suffice to present the variety intrinsically available in the domain. Detailed commentary on these gridfonts and their styles follows in the next section.

1.5 Spirit: Typefaces

As a preliminary to discussion of typographical style in Letter Spirit, it is helpful to consider some human-designed gridfonts. A general approach to style will be established in Chapter 4; for now, commentary on the 23 sample gridfonts will provide a broad set of examples of what variety people can incorporate into gridfonts. Each gridfont was given a name, and in the figures, they are listed in alphabetical order. Most were designed by Douglas Hofstadter; in addition, there is one by Gary McGraw (Weird Arrow), one by Jim Marshall (Intersect), and one via collaboration by the members of the research group (Sluice). Sometimes the primary designer had advice from others.

Standard Square seems like a logical starting point for the discussion. It may be fair to call it simply the plainest gridfont of the bunch, and perhaps one of the plainest gridfonts possible. However, it is actually quite subtle and unusual. While most of its gridletters employ the 2×2 square motif, include right angles, avoid diagonals, and avoid continuous line segments of length 1 (i.e., the number of quanta appearing end-to-end is usually two or four), absolutely none of these is a universal throughout the gridfont. This may be surprising at first, but the lack of absolute regularity is quite ordinary in gridfonts. As later examples will demonstrate, exceptions to general rules are often extremely convenient, if not downright necessary. Standard Square, however, makes exceptions to the trends noted earlier more often than is strictly necessary. While it may be very difficult or impossible to have all 26 letters fulfill all of those conditions, one could certainly do better if outright coherence were the goal. Square Curl, for example, absolutely avoids all diagonals, and offers ideas on how Standard Square could do a better job of being square... if that were the goal.

The key to understanding Standard Square is to note both words in the title; it is certainly square, but it is also very standard. While 'z', for instance, can be rendered without diagonals (as occurs in Square Curl), this would strain z-ness in order to achieve square-ness. Standard Square deliberately makes each letter a very strong member of its letter category. Squareness is a secondary priority, and because it rarely interferes with letter category, it is rampant throughout the gridfont; without considering things carefully, one might mistake squareness for the goal.

This is an odd kind of style, then — a sort of an anti-style that relegates the stylistic properties of the gridletters to a subordinate concern. The real style (namely, the deemphasis of style) is something that applies on the gridfont level, and not on the single-gridletter level. Letter Spirit does not do a superb job of handling Standard Square, because the program assumes that uniformity of spirit is something important to each gridfont; it does not currently allow meta-level design decisions that alter the relative importance of letter-category strength and stylistic coherence.

Benzene Right is a far more conventional gridfont. Along with Benzene Left, Boat, House, Slant, and countless human-designed gridfonts not shown here, it is a strongly *motif*-oriented font. One specific shape, in each case, the shape of the 'o', is the single overwhelming trait of the gridfont. While most letters cannot incorporate the o-ring wholesale (post-and-bowl letters usually can), one could simply say that the designer's goal in each of these gridfonts was to incorporate as much of the o-ring, in the exact location seen in the 'o', as is possible in each gridletter. In addition, pieces of the o-ring, sometimes translated, rotated, or reflected from the original, are incorporated elsewhere. Letter Spirit is well-tooled to create gridfonts of this kind. It can detect motifs, and it can try to draw them. It prefers motifs (entire or partial, with preference for bigger pieces of the whole) that stay in the same place, can tolerate ones that translate without other alteration, and has the least preference for those that rotate or are reflected. The five gridfonts mentioned above are drawn with exactly this sort of principle at work. Gridletters that cannot bear the entire ring include portions of it, as is possible within the constraints of letter category.

These gridfonts may also have *abstract rules* at work. Benzene Right, for instance, absolutely forbids diagonals that slant backwards (from upper left to lower right). Benzene Left, on the other hand, forbids diagonals that slant forwards.

Hunt Four is nearly a motif-based gridfont, but the designer (Hofstadter) arrived at Hunt Four by very different means. Hunt Four is one in a series of nine gridfonts known as the Hint–Hunt family, in which the sheer quantity of material in each gridletter is increased (in the other direction, decreased) as one moves through the gridfont series. The Hunt series has more quanta per gridletter, and the Hint series fewer. Readability is strained at either end of the series, but quite strong in the middle, where Hunt Four lies. Hunt Four usually enforces its o-ring where it can, but misses a few opportunities to do so, as that was not the designer's primary goal.

Slash also strives for inclusion of a motif, only here it is not the o-ring, and this time, the inclusion of the motif is unquestioning. Every gridletter in Slash has a two-quantum forward-slash diagonal across the central zone. When a stylistic property is forced to occur in every gridletter, and is never relaxed in deference to letter-category membership, it is known as a *hard constraint*. A hard constraint frequently leads to weakened letter category for some of the gridletters in the gridfont. In Slash, this is seen particularly in 'n', 's', 'u', and 'w'. Double Backslash attempts a similar motif — the mirror reflection of Slash's slash, in fact — and usually adds an additional, but softer, constraint of a second backslash elsewhere, parallel to the first. Double Backslash's strong spirit damages letter category also, especially in 'z'.

As was mentioned above, in some gridfonts, motifs can move about, by translation, rotation, and/or reflection. Examples of gridfonts featuring these kinds of motifs include Bowtie, Snout, Square Curl, and Weird Arrow.

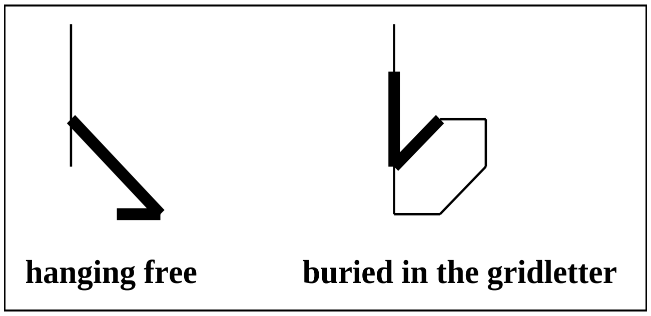

Figure 1.6: Context, as well as shape, may help define a motif.

The idea of a shape motif (depending upon translation, reflection and/or rotation) is present in the styles Checkmark, Sabretooth and Flournoy Ranch, but these gridfonts all have something more to them. Checkmark features the checkmark shape of its entire 'r' in nearly all its gridletters, but this is an underspecification of the style. The context in which the motif occurs is also important. While motifs as those seen in Slant and Slash may be "buried" inside bigger shapes, the Checkmark motif usually coincides with termination of a part; several letters have weakened letter category in the form of unusual gaps and discontinuities ('a' and all the post-and-bowl letters). The smaller a motif is, the more likely it is to be found by chance within any given gridletter, and so the three-quantum motif of Checkmark is rather unremarkable, when found in and of itself. A crucial aspect of this motif is not just its shape, but also the way it is emphasized by surrounding context. Checkmark calls for its motif not only to appear (as on the right side of Figure 1.6) but to appear in a certain way (as on the left side of Figure 1.6). This stronger way of describing a motif is reminiscent of pattern-matching that is sensitive to position of a pattern with respect to the beginning or end of a string. In the UNIX utility `grep`, for instance, one may search for the pattern "tion", and find all strings containing "tion", or search for "tion$", and find all strings ending with "tion". In short, context within a letterform is an important element of motifs.

Sabretooth is another example of this phenomenon; its motif is small and easy to find in many gridletters from other gridfonts. Again, it is not the mere presence, but the *emphasis* of the motif that distinguishes this gridfont. The short squiggle of Sabretooth 'i''s halfpost (as the role is called in Letter Spirit) appears in most Sabretooth gridletters. In the case of Checkmark, the additional distinguishing characteristic of the gridfont was that the motif tended to occur at tips of parts, and the checkmark's tip or tips were left hanging free. With Sabretooth, the motif is often swallowed up inside the gridletter, but it seems to be intentionally placed awkwardly, in order to make it stand out. For example, the Benzene Right 'm' is an elegant gridletter with strong m-ness, and which contains the Sabretooth motif. Sabretooth letters tend not to be such good members of their letter category. Sabretooth is a surly little gridfont! It is not enough to include the motif; the motif must also jolt the eye. Sabretooth 'm', like almost all Sabretooth letters, has the motif in a place that deemphasizes letter-category membership while emphasizing the motif. Just as Standard Square makes a stylistic property of strong letter-category membership, Sabretooth makes a stylistic property of weak letter-category membership. Figure 1.7 captures the distinction between mere inclusion of a motif and inclusion with salience.

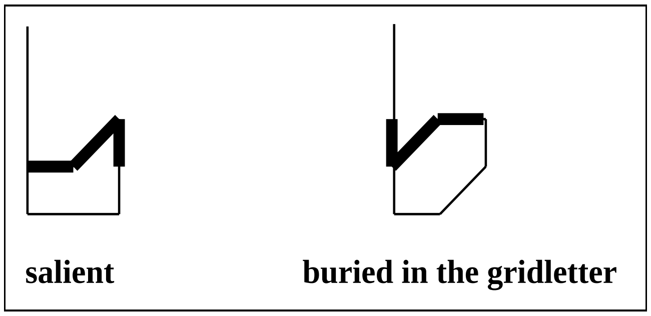

Figure 1.7: A given motif may be more or less salient, depending upon how it is located in the gridletter.

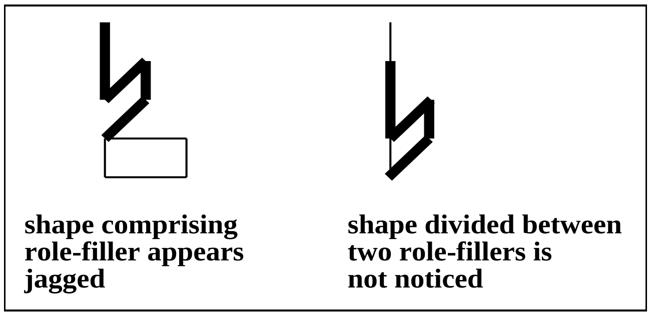
shape comprising role-filler appears jagged
shape divided between two role-fillers is not noticed

Figure 1.8: A given motif may go with or against the natural flow indicated by role-fillers.

Flournoy Ranch plays a similar game. The difficulty in pinning this down is highlighted by the fact that the designer (though years after the creation, not at the time of design) felt that it was hard to pin down exactly what Flournoy Ranch's style was, although it was certainly something [Douglas Hofstadter, personal communication]. Most letters have a point at which three (or four) quanta come together in a T-junction, often, like in Sabretooth, in such a way as to be jarring. The 'a' of almost every gridfont already has a junction where three quanta come together, but the Flournoy Ranch 'a' has a junction where a fourth nub (maybe a serif?) splits off. The style also has a fondness for 45° angles, but, again, it is not just that they exist, but that they blatantly conflict with the letter-category considerations of each gridletter. The 'e' for example, would probably be a stronger 'e' if the final quantum leading to the gridletter's only tip, which points northeast, had just run due east along the bottom. The style here violates expectations of smooth flow in the pathways that one might visually trace through gridletters in plainer gridfonts such as Standard Square and Benzene Left and Benzene Right. Note that there are sharp angles in, say, the Benzene Right 'b' at the upper of the two points where the post meets the bowl. However, the eye does not trace that angle; the post implies one curve, and the bowl another; pathways that cross role-filler boundaries are not noticed. Flournoy Ranch, on the other hand, has jagged paths that cannot be ignored, no matter how one visually traces the letterform. The contrast is between a motif that goes along with the expected flow of a gridletter and one that violates it. Two examples of 45° angles — one that jars the eye by going against the natural flow of its letter and one that does not — are shown in Figure 1.8. The same motif appears in both, but only in the one on the left, where it lies entirely within a single role-filler, does its jaggedness stand out. The contrast bears out the importance of roles as an organizational element in the perception of letters.

Funtnip employs several different tricks. It has a rotatable, reflectable o-ring motif that is adorned with an added serif on the 'o' and that could perhaps be considered a "vestigial" post, as the post-and-bowl letters all grow their posts from the same unusual location as the extra nub on 'o'. Most non-post-and-bowl letters have a two-quantum-wide horizontal segment that is placed somewhere that it is not expected (the middle of 'z', for example) and thus weakens the letter category of those letters. Among the few gridletters not graced with one of those stylistic properties are 'f', 't', and 'x', which all have the unusual property that one quantum is shared between two role-fillers. The combination of these three characteristics (usually just one per gridletter) results in a gridfont with reduced legibility, but with a distinctive style. Is it coherent? By the above analysis, it may seem to be three partial gridfonts cobbled together, with each gridletter employing only one of the three approaches. The gridfont seems more coherent than that, though, because sometimes more than one of the three occurs within a single letter (e.g. 'b'). Moreover, the three have some visual similarity. All three make the letter poke outward in odd places, whether by means of an extraneous tip or a 45° angle jutting out — something that people can see as pointy enough to act as a tip, even though it is blunter than an actual tip.

Sluice, Shorts, and Intersect are each defined, in part, by one property that marks every letter (where possible). In the case of Intersect, it is a pair of quanta that cross each other (which can obviously be considered a motif). Sluice divides each gridletter into two unconnected parts. Shorts makes one (sometimes more than one; in the case of 'o' and 'x', none) line segment in each gridletter shorter than one might expect; most gridletters in Shorts are gridletters from Standard Square with a quantum or two edited out. Beyond these properties, Intersect, Sluice, and Shorts also have additional stylistic traits that characterize each. For Shorts, there is a strong constraint forbidding ascenders and descenders from using the topmost and bottommost portions of the grid; this may also be thought of as a consequence of the inherent shortness of parts already mentioned. The traits that define these gridfonts are widespread, but not absolute; each of these gridfonts has two to five gridletters that do not possess the defining trait. As seen before, the attempt to impose a stylistic property can often detract from letter-category strength, and the designer then chooses — perhaps on the basis of a single gridletter, or perhaps font-wide — between letter and spirit. Some traits conflict with letter category more than others; many of the gridletters of Sluice are nearly illegible, while Shorts (especially) and Intersect are very readable.

Close is also highly legible, but is defined in perhaps the strangest way of any of the gridfonts seen so far. Each gridletter is intended to be very similar to at least one other gridletter, in the sense that each gridletter, by means of either the addition or deletion of one quantum, can be transformed into another member of the gridfont. In principle, this would allow a highly non-coherent gridfont — even thirteen pairs of close matches would fit the bill. As it happens, and as was seen with the last three gridfonts mentioned, a sense of coherence is also applied as a secondary concern; the triangular o-ring is a strong motif throughout. In addition, most posts and tails are short, so that the post-and-bowl letters differ by only one quantum from 'o'. This is a highly abstract and even recursive way to define a style. It is recursive because whether or not a gridletter satisfies the style is a function of the complete gridfont, which is composed of the individual gridletters. One could consider a sizable subset of the gridfont, or perhaps the *entire* gridfont, and still fail to notice the defining trait. This style is based on a relationship between the gridletters unlike any seen so far, and is relatively abstract.

The last gridfont in the sample set, Three-D, is a rather extreme example of bending the rules. Although Three-D is moderately readable, it stretches the very meaning of “gridfont”. Each gridletter here is rendered on the grid, but in such a way as to produce the image of a three-dimensional form, as though the letters were rendered not of one-dimensional line segment quanta, but of squares, with a certain depth into the background. One can easily envision how the three-dimensional shapes implied by Three-D — letter prisms, one might call them — could be viewed from straight-on and would then be proper gridletters, typical of those in ordinary gridfonts. It is these imagined planes, mentally rotated into portrait orientation, and not the line segments actually present in Three-D gridletters, that serve as the role-fillers that indicate letter categories. It is probably safe to say that perceiving Three-D accurately requires the ability to mentally map between two- and three-dimensional shapes. This ability is something that can be taken for granted in people, but is not one of the more obvious abilities that one would consider building into a computer model of letter cognition.

There is no limit to the kinds of foundations upon which a style can be based; a variety of them have come up in the discussion of these 23 gridfonts. The work on Letter Spirit aims to capture a reasonable range of styles by implementing (with routines allowing for perception as well as creation) a number of *stylistic properties* — ones that can define many kinds of gridfont, though not all of the ones listed above. This first implementation of Letter Spirit employs three kinds of stylistic property, upon any combination of which a gridfont can be based.

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

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

The third kind of stylistic property is that which occurs when role-fillers deviate significantly from their corresponding roles; these are called *norm violations*. If a role-filler lacks a particular property expected for its abstract role, that need not disqualify its membership in the category, although it makes it a less prototypical member of the category. A norm violation (e.g., less than normal height) can lend one sort of coherence to a gridfont.

In summary, gridfonts in their full richness employ cross-letter commonalities in a large number of ways, the most straightforward of which are motifs, abstract rules, and norm violations. More complex styles operate on a higher level, and call for the combination of basic stylistic properties, stipulate contexts in which motifs are expected, or modulate the degree to which style conflicts with letter-category considerations. Three-D shows that even more complex kinds of abstraction are possible. Letter Spirit currently focuses on the more straightforward aspects of style, and does not do a very good job of handling complex styles.

1.6 Overview of the Letter Spirit program

The current implementation of Letter Spirit is a program of roughly nineteen thousand lines. When it runs, the action is dominated by three modules — the Examiner, the Adjudicator, and the Drafter. In addition, some other code ties the three modules together.

The Examiner, or, to be precise, the first version thereof, was the subject of Gary McGraw's dissertation. The Examiner takes as input a set of quanta, which presumably constitute a gridletter, and tries to determine which lowercase letter, if any, it represents. Successful recognition of the input as a gridletter also results in the identification of which parts of the gridletter correspond to the abstract roles in the answer's corresponding role-set.

The other two modules are original to this thesis work. The Adjudicator takes the Examiner's output as its input, and tries to identify what style underlies the gridletter. The stylistic properties of the gridletter are used for two purposes. First, it can be compared to the style of those gridletters already considered part of the gridfont in progress, in which case the new gridletter is judged on how well suited it is for inclusion in the gridfont. Second, a set of prospective alterations to the current representation of the gridfont's style are noted, and are utilized later by other parts of the Letter Spirit program, should the gridletter be accepted as a defining member of the gridfont's style, which can evolve throughout a run.

The third module is the Drafter, which, given a letter category and the style of the gridfont in progress, does its best to create a gridletter that is a strong member both of the letter category and of the style. The Drafter may accomplish this by borrowing elements from gridletters already considered part of the gridfont in progress, or by drawing the letter one part at a time, creating parts that correspond to the roles constituting the category.

A sketch of the top-level "glue" holding the modules together describes how the program is intended to function:

1. The user enters from 0 to 25 seed letters.

2. The program inspects the seed letters one at a time by passing them first to the Examiner and then to the Adjudicator. At the end of this phase, the program has built up a representation of the style underlying the seed letters.

3. The program picks a letter category (nondeterministically, but biased towards those that do not yet have satisfactory versions), and directs the Drafter to attempt to render it.

4. The attempt is inspected by the Examiner and Adjudicator. If the Examiner fails to identify it as a member of the intended category, it will be discarded. Otherwise, it will be added to the Scratchpad, the storage place for the new gridfont, along with a combined rating from the Examiner and Adjudicator. Letters that receive poor ratings are particularly likely to attract subsequent attempts.

5. Loop back to Step 3.

The approach captures several essential aspects of creativity, even if it does not model human creativity exactly. The degree to which Letter Spirit successfully models human creativity is the subject of later chapters.

1.7 Overview of the thesis

The remainder of this thesis can be divided into three parts. In Part I, psychological, philosophical, and computational perspectives essential to this work are examined.

This occurs over three chapters, each exploring one of the more formidable problems of intellectual history. The three are related, and ideas developed in Chapter 2, particularly, form the foundation for the rest of the thesis. Each chapter, while overtly concerned with issues of general (and ancient) interest, subtly corresponds to the theoretical framework needed to make sense of one aspect of Letter Spirit: Chapter 2 to categorization (the Examiner); Chapter 3 to style (the Adjudicator); and Chapter 4 to the creative act (the Drafter and the Letter Spirit program as a whole). In Part II, the actual Letter Spirit program and its output are described and discussed. Part III provides a retrospective on this project, evaluating its successes, its shortcomings, and what future directions might be taken.

The breakdown by chapters is as follows. Chapter 2 presents empirical evidence for two modes of letter cognition, which correspond, if loosely, to dichotomies made elsewhere [Sloman 1996]. Chapter 3 is a survey of perspectives on beauty, style, and quality from a variety of fields and subfields, ranging from aesthetics to artificial intelligence. A synthesis of these ideas shows what relevance Letter Spirit has to the hard and unsolved problems in aesthetics. The goal is to illuminate what a working Letter Spirit program can tell us about human creativity. Chapter 4 considers the definition of and nature of creativity, and explores how the level of quality of output that comes from a creative system depends upon certain design properties of the system.

Chapter 5 describes the gridletter-recognizing Examiner and its output. An introduction to the principles behind the Examiner is also an introduction to many shared memory structures and units in the other Letter Spirit modules. Chapter 6 describes the style-recognizing Adjudicator and tests of its work in isolation from the other modules. A description of how Letter Spirit handles style will aid in understanding the gridletter-creating Drafter, which is described in Chapter 7.

It is the interwoven application of the three modules that makes up the full Letter

Spirit program. Chapter 8 presents the program, along with sample output that demonstrates the strengths (and the weaknesses) of the model.

Chapter 9 attempts a global evaluation of Letter Spirit — the contributions of this work as well as the shortcomings that provide directions for future research. Finally, Chapter 10 is a retrospective on cognitive modeling, incorporating the lessons learned during the effort to bring this program to fruition.

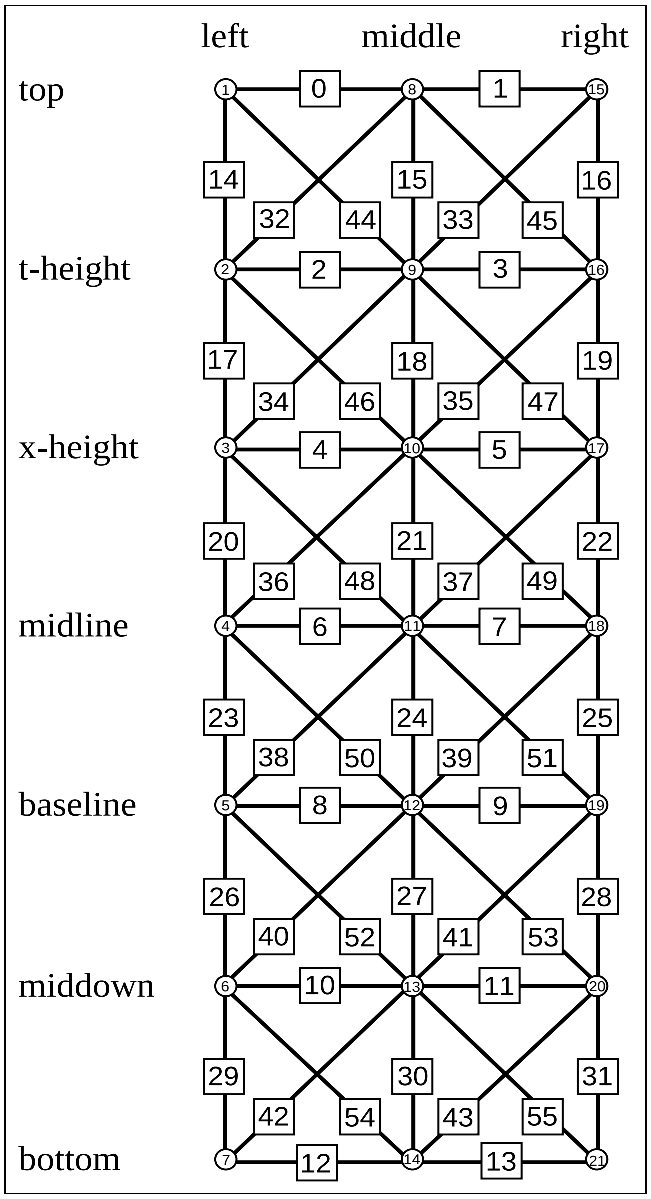

Figure 1.2: Letter Spirit's grid, and certain internal designations.

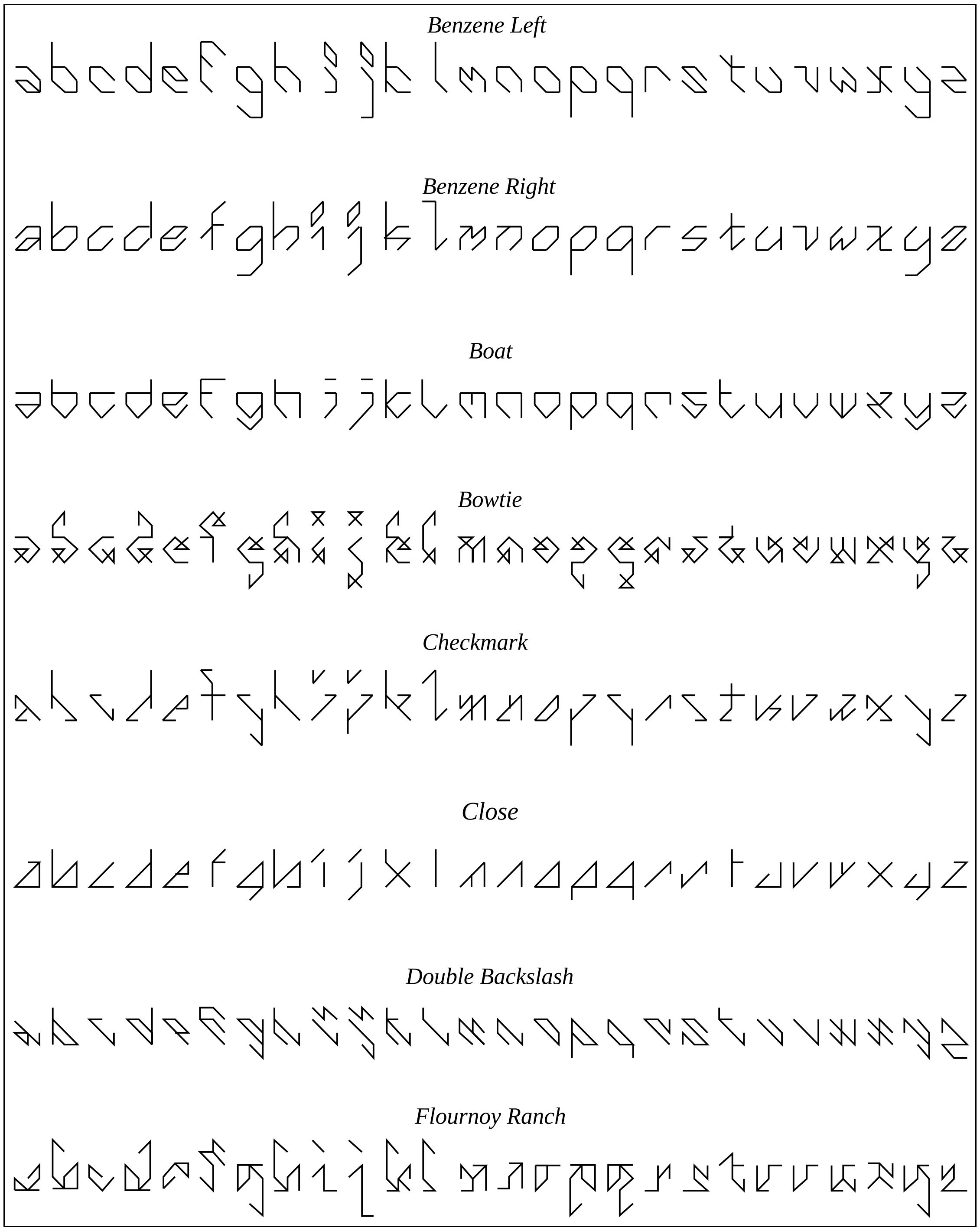

Figure 1.3: Human-designed gridfonts, 1.

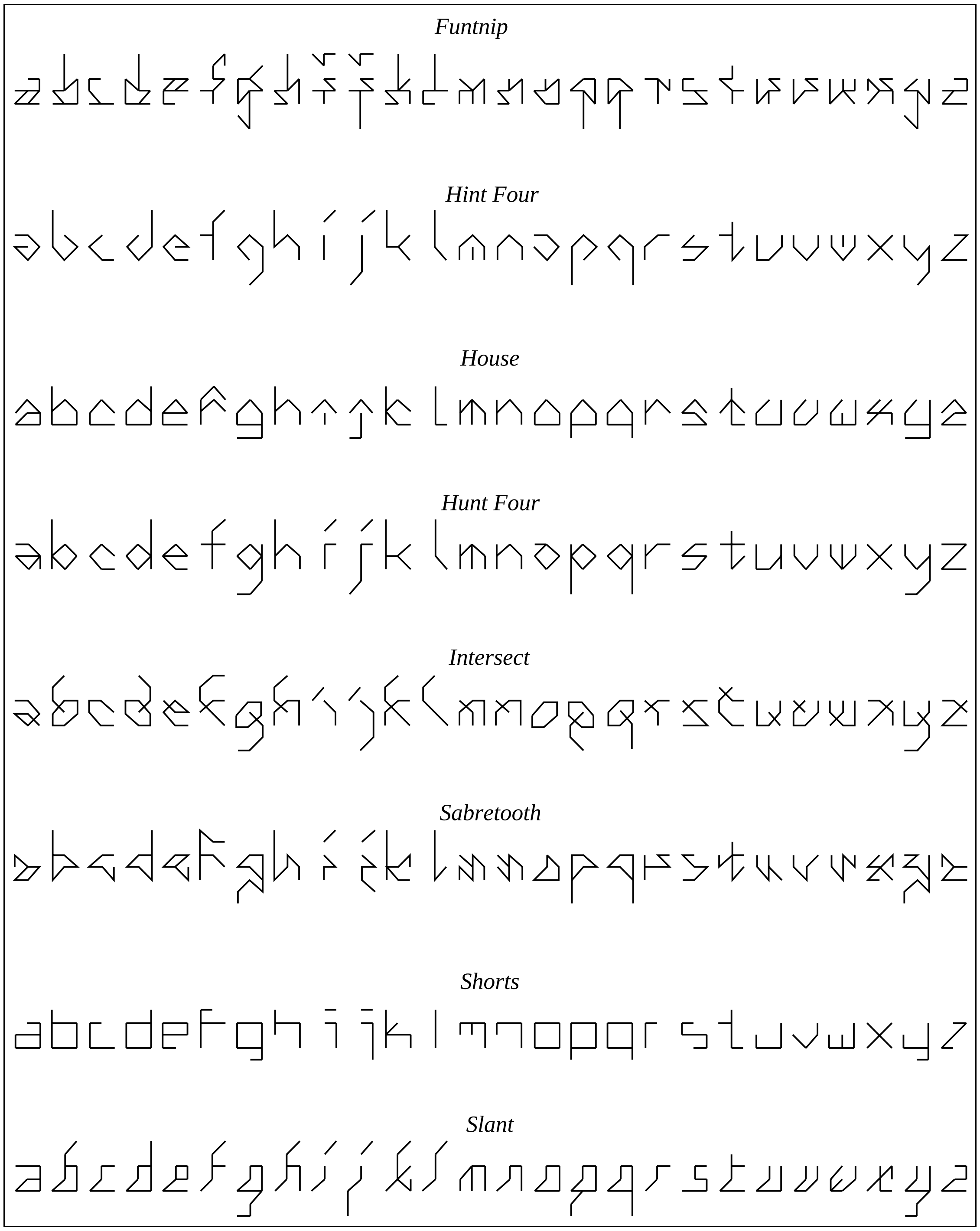

Figure 1.4: Human-designed gridfonts, 2.

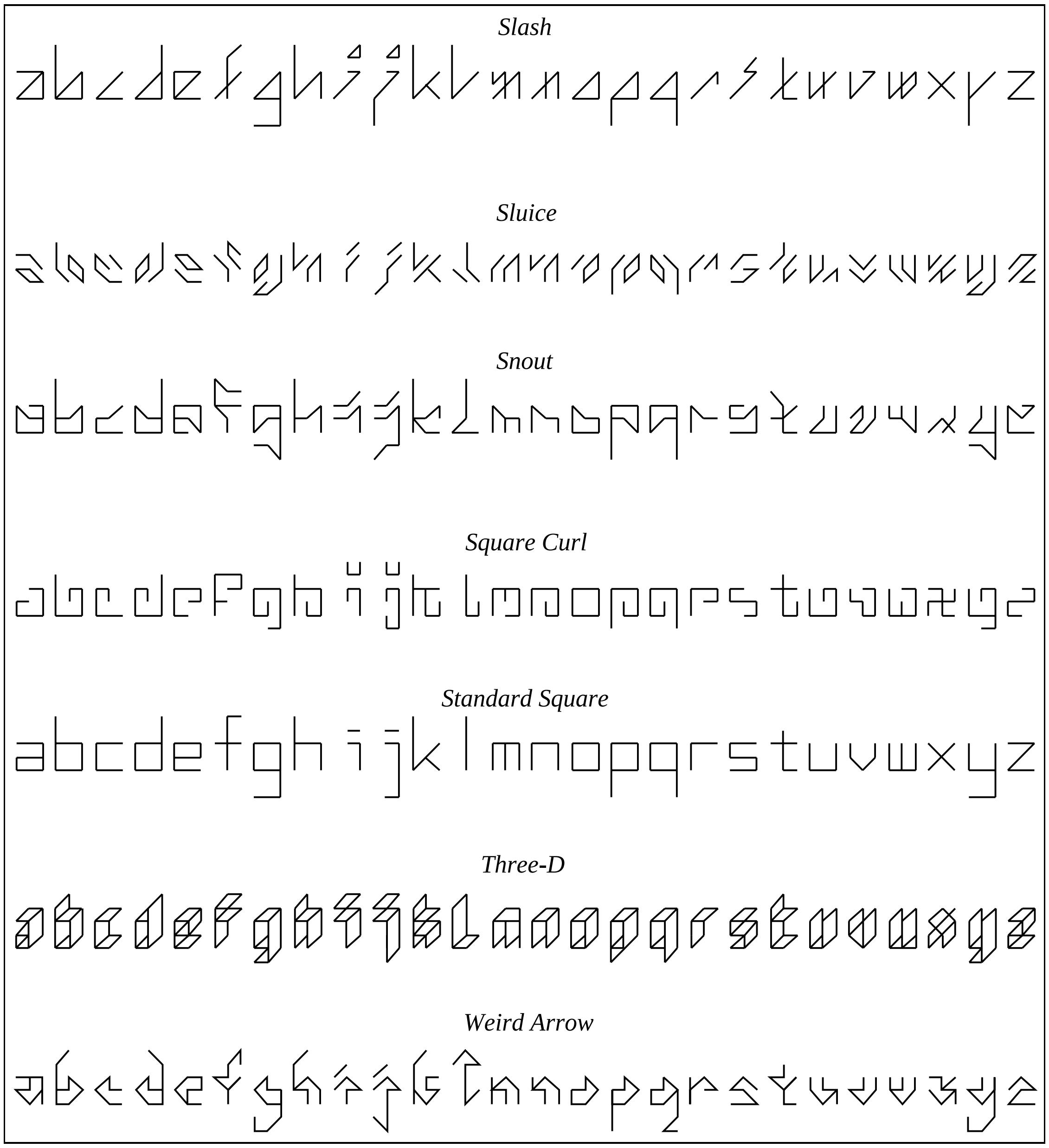

Figure 1.5: Human-designed gridfonts, 3.

CHAPTER TWO

Letter Cognition: Two mechanisms

2.1 Introduction

FARG models have always tried to capture the way mental activity operates at the border between the cognitive and the subcognitive. A central tenet is that the relatively directed and purposeful behavior seen on a high level can be the emergent result of many "small" and less-purposeful events on a deeper level [Hofstadter 1985; Hofstadter and FARG 1995]. The idea that cognition may involve activity on at least two different levels goes back to antiquity, but the overall picture has remained rather sketchy to the present day. The issues seem to be easier to negotiate when considered in a microdomain, where the investigation can be kept grounded in specifics.

Psychology experiments probing human letter perception have been part of the Letter Spirit project since 1993. In [McGraw 1995], the results of experiments on human subjects' perception of gridletters were used to make broad comparisons showing the overall similarity in behavior between the Examiner (as then implemented) and the human letter-recognition facility that it was intended to model.

The goal of this chapter is to describe a relatively detailed model of human letter perception that incorporates two distinct mechanisms, with a basis in the two levels of cognitive activity, which have previously been only hazily defined. The model is supported by experiments with human subjects, and provides a framework that is consistent with the previous literature in the area. Establishing the model early on is vital to a number of goals for later portions of this thesis. The Letter Spirit program's mechanisms and submechanisms will be compared to corresponding aspects of the model on an item-by-item basis in Chapters Five through Nine. Before that, in Chapters Three and Four, the framework will be used as a guide in attacking two thorny issues that lie at the interface between art and science, and each of great relevance to the Letter Spirit project.

2.2 The contrast

There are two kinds of visual memory: one when you skillfully recreate an image in the laboratory of your mind, with your eyes open... in such general terms as "honey-colored skin," "thin arms"... and the other when you instantly evoke, with shut eyes, the objective, absolutely optical replica...

— Vladimir Nabokov, *Lolita*

The study of cognition has often led to a pair of alternative models that contrast in ways that have become familiar. This contrast has been the object of considerable controversy and scrutiny, and vaguely implies a sort of continental divide of cognitive processing. A good starting point for discussion is the approach that Steven Sloman has taken towards characterizing what he calls "two systems of reasoning" [Sloman 1996]. Table 2.1 presents the contrasts that Sloman considers definitive of the two systems.

This table seems to capture the gist of a frequently remarked-upon dichotomy that ranges across a broad variety of cognitive activities. The dichotomy is, however,

	Associative system	*Rule-based system*
Source of knowledge	Personal experience	Language, culture, and formal systems
Nature of representation		
Basic units	Concrete and generic concepts, images, stereotypes, and feature sets	Concrete, generic, and abstract concepts; abstracted features; compositional symbols
Relations	(a) Associations (b) Soft constraints	(a) Causal, logical, and hierarchical (b) Hard constraints
Nature of processing	(a) Reproductive but capable of similarity-based generalization (b) Overall feature computation and constraint satisfaction (c) Automatic	(a) Productive and systematic (b) Abstraction of relevant features (c) Strategic
Illustrative cognitive functions	Intuition Fantasy Creativity Imagination Visual recognition Associative memory	Deliberation Explanation Formal analysis Verification Ascription of purpose Strategic memory

Table 2.1: Sloman's two types of reasoning.

difficult to characterize very well. Sloman's effort to do so is comprised of distinctions that are nebulous and vague because they try to capture such a wide range of human behavior with a few pithy labels and phrases. Even the accompanying article, though it expounds upon these distinctions at greater length, offers only a hazy characterization of the overall dichotomy. This chapter attempts to make a more precise characterization of just such a contrast by focusing on one task — the categorization of letters. This chapter reports an investigation along these lines, in the hope that the focus on just one task may make it possible to produce more definitive evidence of the psychological reality of the dichotomy, and to describe the dichotomy more precisely.

Something resembling Sloman's set of contrasts surfaced in analysis of the first FARG experiments on human letter perception, when the results in [McGraw et al. 1994] led to conclusions at odds with those of some previous studies, even though all the studies in question were based on error-making in letter-categorization tasks. There were also suggestive differences between the errors on trials in which subjects responded more quickly and those in which responses came more slowly. Additionally, it was noted that intercategory similarity as derived from a categorization task was not quite the same as that derived from similarity ratings [Podgorny and Garner 1979]. All of these things suggested that more than one type of letter perception might exist.

[Rehling 1996] found evidence for this in a comparison of previous studies that had, like [McGraw et al. 1994], generated error matrices derived from categorization of lowercase letters. Papers such as [Bouma 1971; Townsend 1971; Geyer 1977] had each argued for their own proposed mechanism, but all those models shared a similar kind of representation at their core. All of these models used a two-dimensional image map (like a retina) of the letter as their only representation. Others [Blesser et al. 1973; McGraw et al. 1994; Sanocki 1986], however, have argued for structured, hierarchical representations, in which components (loosely conforming to the strokes one would make in drawing the letters) are the basis of letter perception (and therefore categorization). Moreover, it was shown that mere image-map (or "flat") representations correlated relatively poorly with the data [McGraw 1995]. Thus, it seemed that two broad camps within the literature were at odds. An argument first offered in [Rehling 1996] (and summarized later in this chapter) resolved the conflict by proposing that the two camps were actually studying two different mechanisms.

These two mechanisms have a great deal in common with those identified by Turvey [1973]. By means of a series of experiments testing the ways in which distractors may mask stimuli by occurring just before or just after them, and thus prevent accurate perception, Turvey concluded that there exist (at least) two kinds of process involved in visual perception: “peripheral” processes, which do not require that attention be directed at the stimulus (namely, the stimulus is kept outside of foveal vision), and “central” processes, which do involve attention. Turvey concluded that peripheral processes act more quickly than central processes, and that central processes are contingent upon the outputs of peripheral processes. He hypothesized that masking due exclusively to peripheral processes dominates behavior in the first few tens of milliseconds of a perceptual event and that there is a transition after that time so that masking due exclusively to central processes dominates thereafter.

The rest of this chapter is intended to show more clearly that two subsystems are indeed at work in letter categorization. The immediate goal is an account of two mechanisms — ones that contrast in many of the ways implied by Table 2.1 and by Turvey — each of which contributes in letter recognition. Unlike the work in [Rehling 1996] and [Turvey 1973], the approach taken here will use a single experimental design with only one independent variable to illustrate the effects of both kinds of mechanism. The results will also suggest some reasons why the two may coexist; a discussion of how they may interact will lead to a simple process model.

2.3 Studies of letter categorization

The perception of alphabetic figures is an activity that requires less complexity than does vision in general. Letters do not represent real-world objects, and they do not have norms for color, depth, texture or absolute size. As such, the variety in letters is isolated to their forms, and this makes them appropriate stimuli for tests of visual categorization that are primarily concerned with form.

In the many accounts that have been proposed for letter categorization, one finds a large number of different mechanisms, but only a small number of underlying representations. The two kinds of representation invoked most often are those that suppose an image map with no structure but that of the Cartesian plane, upon which sensory processing operates directly [Luce 1963; Bouma 1971; Geyer 1977; Townsend 1971]; and those that employ a structured representation [Sanocki 1986; McGraw et al. 1994], with a level of organization below that of the letter level, in which roles for substructures such as bowls and posts must be identified before full letter categorization can take place. The former kind, because they have no hierarchical structure, will henceforth be called "flat" representations, and the latter, "structured" representations. A third paradigm for letter categorization, which is based upon feature lists [Gibson 1969; Keren and Baggen 1981], is also addressed briefly below.

There is a bit of a paradox in that the literature contains evidence for models of letter categorization that appear to be incompatible with one another. The resolution to this lies in the fact that the papers offering evidence for one kind of model seem to be studying a different phenomenon than those offering evidence for the other — despite the fact that all claim to study the same thing. A look at the experimental methods that are used by the two camps explains the apparent contradiction. Any experiment that seeks to produce an error matrix based on letter categorization must somehow degrade the stimuli, or otherwise, there will be no errors to study. Different ways of degrading the stimuli tend to indicate models with different underlying representations. Specifically, those papers that proposed flat representations degraded the stimuli in ways that prevented the subjects from attending to the structure of the letters, either by moving the stimuli well outside the fovea, or by reducing the apparent size of the stimuli beyond the threshold of easy recognizability, or, most interesting, by shortening the stimulus exposure to roughly 100 ms or less. In contrast, the proponents of structured representations degraded stimuli by presenting letters rendered in diverse and unusual styles [McGraw et al. 1994; McGraw 1995], or by mixing letters and nonletters in crowded displays, or by blanking out portions of letter stimuli [Sanocki 1986]. Experiments supporting structured representations, in all cases, kept the stimuli available for a long time (in fact, until the response). This is also true of experiments involving similar but non-alphabetic stimuli used to support the idea of structured representations [Palmer 1978].

A solution to the dichotomy between the two models is to propose that the perceptual system has (at least) two mechanisms, either of which can carry out letter perception, and that the choice of which one determines behavior depends — at the very least — on the chronology of the perceptual act. The two hypothesized mechanisms are a fast mechanism, using flat representations, and a slower but more accurate (and preferred) mechanism employing structured representations. The observations of all the studies can be explained by supposing that the structured-representation mechanism's answer is preferred whenever both answers are available (namely, whenever stimulus presentation was sufficiently long). In order to verify this, a letter-recognition experiment varying only the duration of stimulus presentation was undertaken. Except for this variable, the experiment was very similar in design to that of [McGraw et al. 1994]. The test of which theoretical model best explained a given set of data is explained in the next section.

2.4 Theories of intercategory proximity

Intercategory errors provide a valuable measure of intercategory distance. Consider the following errors that might be found in a newspaper:

As this illustrates, errors can identify the type of representation that was involved in the production process. The above errors are relatively unambiguous in their origin. With alphabetic intercategory confusions, things will not be quite as clear-cut; a single error alone will never pinpoint which type of representation was involved. However,

Intended	*Actual*	*Error due to...*
barn	bam	Visual similarity
their	there	Phonetic similarity
crust	cryst	Keyboard layout similarity
Soviet	Russian	Semantic similarity

Table 2.2: Error-making can reveal what kind of underlying representation type was involved.

trends and correlations in the entire set of 325 possible bidirectional intercategory confusions yield a kind of diffuse fingerprint of the underlying representation.

The two kinds of representation most frequently invoked by models of letter recognition are both examined here. The flat-representation model is based on the sum of two calculations of intercategory distance — one involving the position of "material" within a letter, the other involving the position and presence of tips. Both calculations were based on category prototypes derived from many examples of each letter category, with many examples for each category. The examples, like the stimuli in the experiment, were gridletters on the usual Letter Spirit grid.

The prototype for a letter category in the flat model is based in large part on the "blurred-quanta" prototype made from all the example gridletters belonging to that category. In an ordinary gridletter, a quantum is either turned off or on. In the blurred-quanta prototype, the degree to which a quantum is on can be any value between 0 and 1; the prototype's on-ness value for any given quantum is simply the proportion of the examples in the given category that have that quantum turned on. The intercategory distance between two blurred-quanta prototypes is calculated by taking the sum of the differences (that is, the absolute values thereof) over all 56 quanta. In addition, the flat model was augmented with blurred-tip prototypes, something very like the blurred-quanta prototypes, but based on the locations of tips, rather than turned-on quanta. This employed a slightly coarser spatial representation — that of the twelve 1×1 squares in the grid. For each category, a tip prototype was calculated by counting the average number of tips that the example gridletters in that category had in each of the twelve squares. The best fit to human data (in the fastest stimulus- exposure conditions) for this enhanced flat representation model came from a calculation of intercategory distance that added blurred-quanta distance to 0.75 times the blurred-tip distance. The integration of two flat representations into one might seem to disqualify the model as truly “flat”. However, it is non-hierarchical in the sense that the two measures of intercategory distance are merely summed together after each measure is applied separately to the entire figure. It has been shown [Treisman and Gelade 1980] that tips are primitive features that do not require attention to be perceived. Cave and Wolfe [1990] discuss how multiple feature maps — each map corresponding to the same visual area, but each isolating a different type of feature — can be handled by the perceptual system as though the feature maps were all primitive and at the same level. This kind of interaction between multiple kinds of feature maps lacks the binding-together of multiple features into an object perceived within the image map that is the hallmark of structured representations.

For the structured-representation model, each category’s prototype was defined in terms of constituent parts: closures and curves of various types resembling those in [Sanocki 1986] (although Sanocki was concerned with uppercase letters rather than lowercase). These prototypes are simplifications of the role-based structures that Letter Spirit uses to represent letters (see Chapter 1 and Chapter 5). They are a bit simpler, in order to facilitate direct intercategory comparison, and have thus been called “proto-role” representations, to distinguish them from those implemented in the program. What role and proto-role representations have in common are the essential properties of structured representations — that a letter whole is defined in terms of parts into which a whole can be divided. A whole is defined as a set of abstract components and norms for the way the components interact. The components, in turn, are defined in terms of multiple underlying properties specifying size and shape. Table 2.3 shows eight letter categories in terms of the structured (proto-role) representations.

Proto-role sets	
a	closure with sharp left hook attached at NE
b	closure with straight post attached at NW
f	vertical cross with top hooking gently right, straight bottom, arms pointing W and E
j	vertical staff hooking right below and having a dot above
l	vertical staff
o	closure
t	vertical cross hooking gently left below, straight top, arms pointing W and E
y	NE-pointing staff with arm pointing NW protruding from middle

Table 2.3: Examples of the structured representations of letters.

In the structured-representation model, intercategory distance between two categories is defined by how easily one category's prototype may be transformed into the other's. The transformation may take whatever deformations are required: changes in how curvy a proto-role is, the addition or deletion of a proto-role from a category's proto-role prototype, or the rotation of the entire prototype. Each of these types of deformation was assigned a weight according to its estimated impact, and intercategory distance is calculated by the sum of the deformations needed to make the transformation between two categories' prototypes. Five parameters (the weights) were adjusted so as to produce a good fit between the structured-representation model and the data collected by [McGraw et al. 1994].

As was noted earlier, there are also models that base categorization upon feature lists. In this paradigm, a set of features, usually boolean in nature, is used to create the category prototypes. Each category's prototype is the vector of 1's and 0's that indicates whether or not each feature should be expected in instances of the category. Recognition in this model takes an input and computes the vector of 1's and 0's for it, then returns the category prototype that offers the best match (in terms of Hamming distance) to the input's vector. An attempt to create a theoretical error matrix for two feature-based models in the literature [Gibson 1969; Keren and Baggen 1981] led to the realization that these models were too rigid to apply beyond the stimuli for which they were intended. Those two feature lists, which were intended to distinguish uppercase letters, fail to distinguish the lowercase letters even in principle, because when they are used to create prototypes for the lowercase letters, some pairs of lowercase letters wind up with identical sets of features. No theoretical error-matrix of reasonable quality could be formed on the basis of either feature list. Although the originators of these lists admit as much, this shows that short feature lists lack generality, which is problematic for a prospective model of perception. Perception based upon features is nonetheless a worthwhile idea, and has connections to other parts of this thesis. The theory behind it will be part of the discussion of higher-level perception in Chapter 3, and recognition by feature lists is similar to the way that the Examiner recognizes *roles* within gridletters, as is detailed in Chapter 5. However, the problems with them as a general model of categorization prohibits their comparison side by side with the other two kinds of representation studied in detail here.

[McGraw et al. 1994] compared their data (as well as that of several computer models of letter recognition) to the flat- and structured-representation theoretical models. They found that human performance had a very high correlation with that predicted by structured representations (calculated with off-diagonal error matrices, $r = 0.88$), and a substantially weaker correlation with the flat representations ($r = 0.37$). In light of the evidence that other experimenters had gathered human data that correlated better with the flat representations, a subsequent experiment was performed, whose goal was show that the two kinds of representation could *each* provide the best fit to human data, depending upon manipulation of the duration of stimulus presentation.

2.5 Experiment: Dual mechanisms

In order to conduct a single investigation bearing out the conclusion of [Rehling 1996] that two mechanisms, each capable of letter recognition, might be distinguished by their speed, the experiment of [McGraw et al. 1994] was repeated with a new independent variable — namely, the duration of stimulus exposure.

2.5.1 Method

Stimuli

A black grid was displayed against a white background on a Macintosh screen. The 21 vertices were drawn with a unit distance of 1.31 cm, making the entire grid 7.86×2.62 cm. A solid square with sides of 0.13 cm was drawn at each vertex. The stimulus was presented for a set duration, then replaced with a mask consisting of every line segment possible in the grid. Subjects were seated, but allowed to maintain whatever distance from the monitor they found best. Samples of gridletters that served as the stimuli can be found all through this thesis.

Procedure

A data set of gridfont characters was shown to each subject. The tokens were presented individually, with the screen blank for two seconds between each response and the next stimulus onset.

Subjects were told that they would see a series of stimuli, each of which was intended to represent a lowercase roman letter. Responses were to be rapid keyboard keypresses that identified the stimulus as one of the 26 lowercase roman letters.

Each subject was presented with an entire database of 545 diverse gridletters, representing each lowercase letter category many times in many styles. Each stimulus was presented for a duration chosen randomly from 17 ms, 50 ms, or 117 ms.

The sequence of stimuli and the choice of duration for a given trial were completely randomized over the subject pool.

Subjects

38 subjects were drawn from the subject pool of Indiana University undergraduates.

2.5.2 General results

For each of the three durations, an intercategory error matrix was generated. Then the matrix was correlated according to bidirectional off-diagonal values with each of the two theoretical matrices. For each correlation, the theoretical matrix was tuned so that its rate of predicted correct answers (the mean down the diagonal) matched the subjects' performance. Results, including each condition's Pearson correlation with each theoretical model, are presented in Table 2.4.

Stimulus duration	*Accuracy*	*Flat correlation*	*Structured correlation*
17 ms	41.8%	0.597	0.509
50 ms	64.8%	0.527	0.615
117 ms	71.8%	0.535	0.686

Table 2.4: Correlation with models vs. stimulus duration.

This was the third attempt to obtain such a result. Two earlier experiments of similar design were carried out with intervals of 100 ms, 400 ms, and 800 ms; and a followup with intervals of 33 ms, 67 ms and 800 ms. It proved necessary to make stimulus durations very short in order to disable the use of structured representations. Table 2.4 shows only the data from the third version of the experiment, but interesting effects based on the first two versions are discussed later.

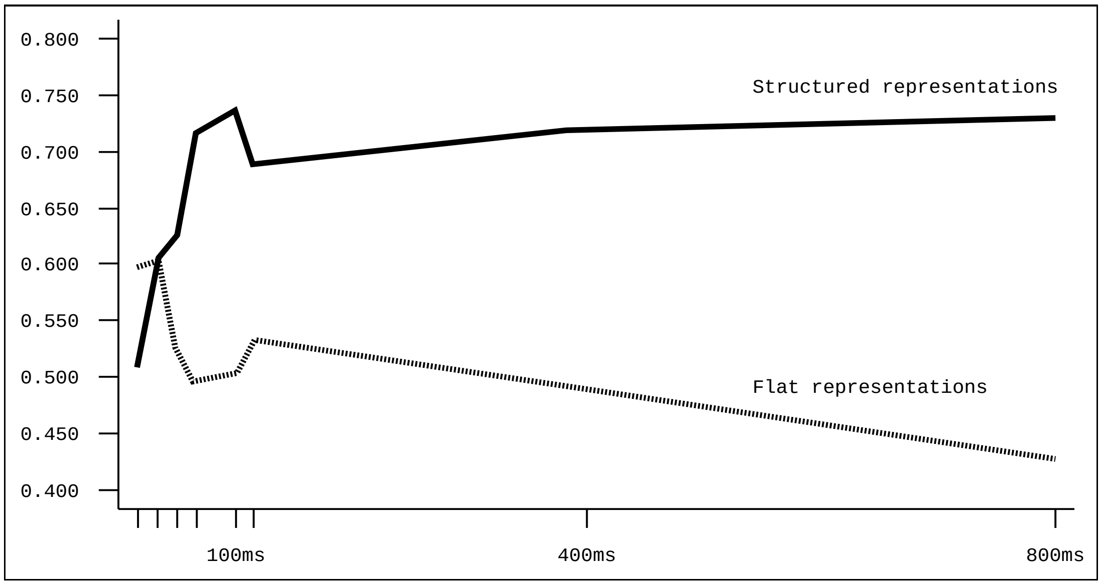

Figure 2.1: Flat representations dominate short stimulus durations; Structured representations, longer ones.

2.6 Interpretation of the results

2.6.1 Really two mechanisms?

The data show that the character of the errors made, as a function of stimulus exposure duration, shifts markedly in the vicinity of 50 ms. At some stage between stimulus and response, there coexist in the mind (at least) two different ways of representing the stimuli. Does the existence of two kinds of representations automatically indicate two separate mechanisms? This seems to be an empty distinction, not dependent on any details of the actual structures and processes underlying the event.

In Figure 2.2, the bold lines indicate actual cognitive mechanisms and structures, and the dashed lines indicate delineations made by hypothetical cognitive scientists deciding what to call a “mechanism”.

One might protest that what seems to be two mechanisms could be just one, which differs, somehow, in its behavior by a matter of degree along a continuum.

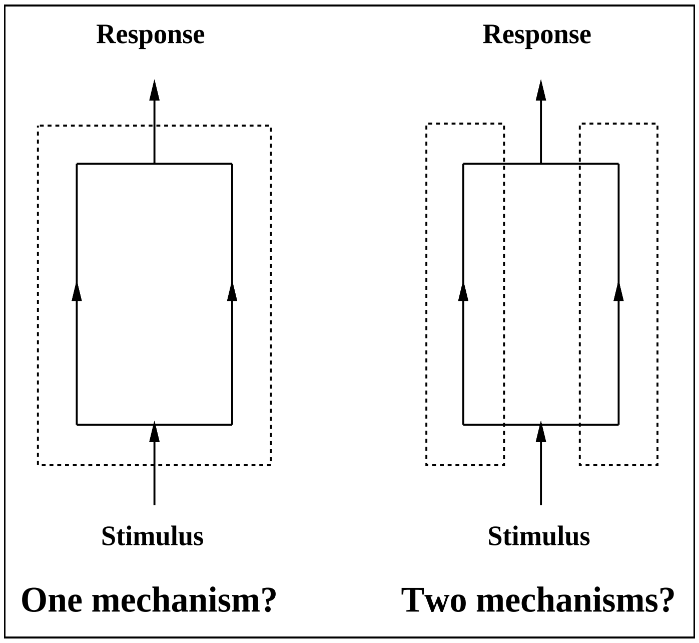

Figure 2.2: The distinction depends upon the stance the outside observer wishes to take.

In this case, however, decomposition of a whole into parts is an essential step, and decomposition seems to be a discrete step that does not allow for gradation along a continuum. Overall behavior in any given circumstance may be influenced by both mechanisms, in varying proportions. The range from about 33 ms to 100 ms seems to show how influence gradually shifts from one mechanism to another. A similar shift in behavior depending upon time-course in a different categorization task is explained by [Goldstone and Medin 1994] as the changing behavior of a single mechanism. However, the delineation between the two mechanisms proposed here is quite sharp. The flat-representation mechanism categorizes the percept as a whole. The structured-representation mechanism discerns parts within the whole, and bases categorization upon this. It does not seem possible that processing acts on some level between that of whole and parts, because it is logically impossible that there *is* such a level — anything below the whole *is* a part. The whole, so to speak, cannot be a little bit pregnant.

Another way to claim that the two mechanisms are (or might be) the same would be to posit that the processing of subunits in the structured-representation mechanism occurs in the same manner as processing of wholes does in the flat-representation mechanism. This is also an empty claim; the models are still distinguished by the fact that one mechanism categorizes structures on a level below the whole, and the other does not. Contrast this with the inclusion of tip features in the theoretical flat model used earlier in this chapter. That did not categorize any elements on a level below the whole — it merely utilized two feature maps (letter "material" and tips) to categorize the whole, without invoking any constituent parts of the whole. The rest of this thesis will take the experimental results as an unambiguous demonstration of the existence of two mechanisms in human perception of letters.

2.6.2 Influence: Divided how?

It is clear that there is a shift, as stimulus duration varies, in how much influence the two mechanisms have. It is not immediately clear how extreme that shift is. The range in correlations in Table 2.4 is relatively subtle — all are between 0.5 and 0.7. This might lead to the conclusion that the shift in influence is also subtle, and that both mechanisms are quite active in all conditions, but a bit less so in some. This does not, in fact, follow. It is possible that correlations of about 0.5 are misleadingly high, and correspond to little (or no) influence of a mechanism in a given condition.

The reason for this is that the two matrices are not orthogonal. They are, in fact, highly related; the correlation between the flat and structured theoretical matrices themselves is 0.452. As a consequence, if each theoretical matrix were a superb model of the related mechanism in the human perceptual system, and if the two mechanisms were, in fact, mutually exclusive in their activity, correlations of almost 0.5 would still be automatic in all conditions.

The truth probably lies somewhere between the two possibilities discussed above.

The theoretical models are certainly not perfect renditions of the human mental phenomena they are meant to model; neither is it claimed that the precise extent of their influence in each condition is understood. It is very hard to imagine that the influence of the faster mechanism utterly vanishes in the trials with longer stimulus durations. It is conceivable that the slower mechanism cannot act below a certain threshold of stimlus duration, but it is not clear whether this experiment probed below such a threshold or not.

2.6.3 The two mechanisms, in intuitive terms

The two mechanisms differ in character in ways that approximate Sloman's two types of reasoning (Table 2.1). In trials with the briefest stimulus durations, responses are best predicted by the mechanism involving flat representations; in slower trials, by the mechanism involving structured representations. The correlations reported earlier show a relationship between the two models and human behavior, but specific examples of errors that bear out this relationship are particularly informative.

Similarity of letter categories in terms of the structured representation is often (but not always) found where there is similarity in terms of topology; any deformation involving a change in topology is considered expensive in the calculation of category distance in the structured-representation model. Therefore, the model is relatively strict about seeing topology maintained. In contrast, the image map of the flat-representation model primarily reflects where letter "material" is placed on the display, with less emphasis on relations between and organization of substructures within that material. As a result, the flat-representation model is tolerant of near misses in the contact or lack of contact between parts. The contrast in how sensitive the two models are to topology means that the performance of the models can be distinguished from one another by how often they make errors depending upon topology. Some of the top human errors appear in Table 2.5, ranked in order of frequency in the 117 ms condition. These are bidirectional error rates: *xy* is the mean of the rate at which 'x's are mistaken for 'y's and 'y's for 'x's. The error rates for those same confusions in the other experimental conditions are also presented.

	17 ms	*50 ms*	*117 ms*	*Flat*	*Structured*
uv	0.110	0.1398	0.1659	0.0931	0.2714
ij	0.094	0.1219	0.1242	0.0345	0.1946
gq	0.134	0.0970	0.0901	0.1037	0.2142
hk	0.127	0.0780	0.0616	0.0652	0.0558
pq	0.039	0.0475	0.0526	0.0247	0.0063
xz	0.088	0.0556	0.0494	0.0289	0.0189
it	0.056	0.0430	0.0488	0.0407	0.0015
ae	0.027	0.0276	0.0487	0.0525	0.0000
nr	0.051	0.0526	0.0478	0.0873	0.0924
sz	0.042	0.0398	0.0453	0.0392	0.0000
vw	0.065	0.0767	0.0441	0.0481	0.0538
il	0.061	0.0477	0.0426	0.0313	0.1621
lz	0.016	0.0359	0.0421	0.0059	0.0047
mn	0.089	0.0448	0.0417	0.1174	0.0585
kx	0.058	0.0491	0.0411	0.0181	0.0603
ry	0.042	0.0356	0.0370	0.0050	0.0521
fi	0.054	0.0469	0.0362	0.0291	0.0015
jy	0.069	0.0469	0.0348	0.0021	0.0017
ao	0.034	0.0212	0.0337	0.0563	0.0341
yz	0.045	0.0210	0.0336	0.0034	0.0583

Table 2.5: Common errors, in experimental conditions and as predicted by theory.

What is remarkable about the data in Table 2.5 is that it shows that several errors are more prevalent in the conditions where subjects have longer access to the stimulus! This contradicts a naive but natural expectation that one might have regarding this task: that having extra time to consider the stimulus could only increase accuracy. This result is all the more telling because the errors that increase with greater stimulus duration — *uv*, *ij*, *pq*, and *ae* — involve two for which the confused letter pairs' category prototypes are topologically similar, and therefore are predicted to be more easily confused by the structured-representation model than by the flat-representation model. Confusions *uv* and *ij* are predicted by the theoretical models to show this

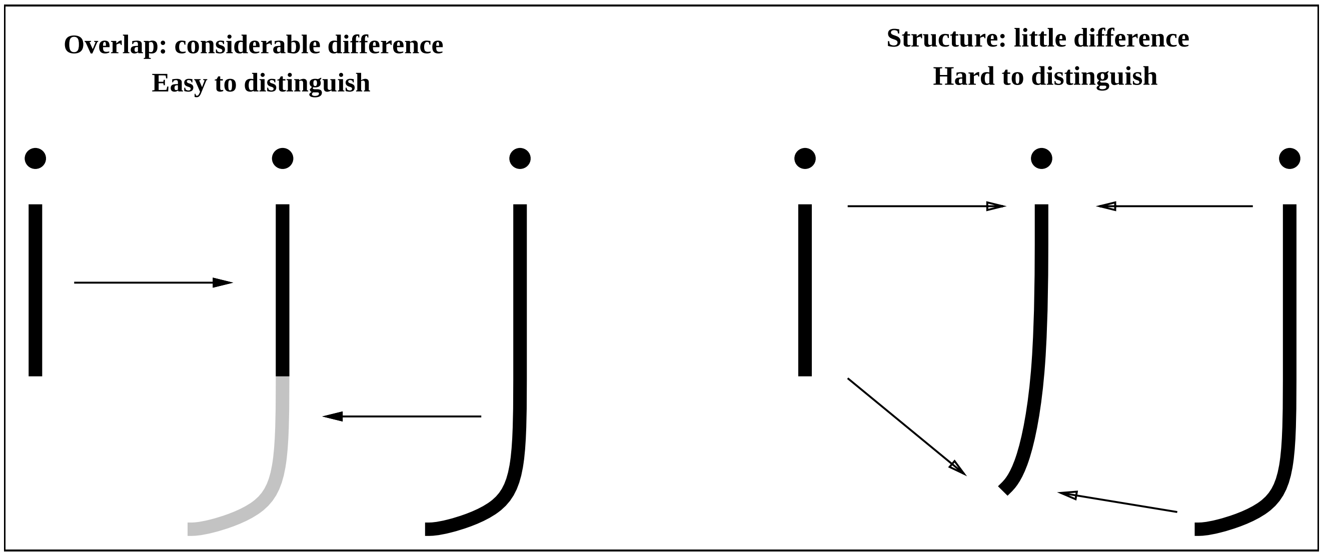
Overlap: considerable difference
Easy to distinguish
Structure: little difference
Hard to distinguish

Figure 2.3: Two categories can be considered very similar or fairly different, depending upon the mechanism behind the comparison.

effect.[1] Figure 2.3 illustrates how 'i' and 'j' are quite similar in terms of structure, but have sizable difference in terms of overlap when they are superimposed onto one another. The experiment shows that people are seemingly obliged to use the structural means of categorization when stimulus duration is long, even in that minority of situations when it is less accurate than categorization by means of flat representations, based on degree of overlap.

Some errors that are relatively common at 117 ms but that are not predicted by the structured-representation model can be explained by sundry "noise" factors. Some of the following are likely to be involved: Categorizations based on capital letters (one 'j' resembled a capital 'S'); subjects' ignoring the central zone/ascender/descender

[1]Confusions *pq* and *ae* are not predicted to show such an effect, but do. The nature of those two errors reveals a shortcoming in the structured-representation model. These confusions are facilitated by mental reflection over a line and major rotation (greater than a few degrees) about a point, respectively. The method used to calculate distance in structured representation space makes those types of transformations expensive, but these data suggest that the mechanism subjects use in processing structured representations does allow this as a basic transformation. A modification would make rotations of 180° less expensive than those of intermediate measure; likewise, mirror reflection should be made a relatively minor transformation.

distinctions (one ‘e’ was categorized as a ‘p’ by subjects who ignored the fact that it was located entirely in the central zone); figure-ground reversals (the pointed arch of an ‘h’ could be taken as the two right-pointing arms of a ‘k’). These are doubtlessly factors in letter perception, but they go beyond the simplicity of the structured-representation model.

2.7 Back to the continental divide

The two mechanisms found here differ in ways that approximate those listed in Table 2.1. This section explores Sloman’s contrasts and evaluates the evidence that this investigation produced for each of them. The direct evidence is simply that the mechanisms differ with regard to representation and speed. There is circumstantial evidence, though, for a few other contrasts between them.

2.7.1 Attention

Several of the possible contrasts between the two mechanisms are related to the question of attention, and the results of the experiment are consistent with the consensus account of attention. Attention is commonly assumed to correlate with conscious awareness and with deliberate control [Posner 1995]. Attention offers a kind of enhanced perception of an object, and can be shifted from one target to another, but the effects of that shift do not begin to show up until a certain amount of time, about 150 ms, has passed [Desimone and Duncan 1995; Posner and Petersen 1990]. In addition, at least one attentional mechanism (there may be more than one) seems to pertain to the perception of objects, rather than mere locations within the perceptual field [Vecera and Farah 1994]. The dichotomy between the two mechanisms can be related to attention by means of the following succinct account:

When vision is fixed on the percept, an automatic perceptual facility, akin to

Sloman's associative mode of reasoning, Ullman's categorization by overlap [Ullman 1989], and Turvey's peripheral processes, begins work immediately. It does not require attention or entail awareness of its internal functioning. It quickly produces an output that categorizes the percept. Beginning at the same time, a second mechanism, operating concurrently with the first one, orients attentional resources on the percept, and after some tens of milliseconds of serial processing, constructs a (potentially) hierarchical representation of the percept as an object. This mechanism, akin to Sloman's rule-based mode of reasoning and Turvey's central processes, also permits categorization of the percept, but it works more slowly than the automatic mechanism. In some situations when the structured-representation mechanism has time to act, the two mechanisms will give conflicting answers. When this happens, the structured-representation mechanism's answer is preferred over that of the flat-representation mechanism. In general, any given study of human performance on a given task may be examining only one or the other of two possible underlying mechanisms.

It should be clear that while the flat-representation mechanism does not require attention on the input, its *result* can become the focus of attention. In fact, the term "automatic" from Sloman's table suggests that a flat-representation mechanism might operate all the time, ceaselessly, with its output always available to other kinds of processing. One may thus consider flat-representation mechanisms to be contributing a number of "feature maps" that are the options among which attention may choose ([Treisman and Gelade 1980; Cave and Wolfe 1990]).

This is reminiscent of the design of serial microprocessors. Operations that require little hardware to support them can be implemented on a broad, parallel basis; usually, only the simple read/write/store operations for memory fit this description. More expensive operations are conducted in multiple but fewer locations; the arithmetic/logic unit (ALU) calculates several functions at once, all the time, though the result of only one of them is used in a given clock cycle. The most expensive operations are only implemented in one location; the clock, the input bus, the output bus, the program counter and instruction register fall into this category. This architecture gives the system the ability to utilize any of the various operations, though some of them face the limitation that they can only be utilized in a serial fashion.

The example of numerical addition shows why the approach is beneficial. If memory contains $1,000$ numbers, it would require one million ALUs to simultaneously compute all possible additions involving any pair of those. This architecture would make for a very fast machine, but the amount of hardware that would require is prohibitive. A practical solution is to allow simultaneous access to all $1,000$ numbers in memory (this is cheap in terms of hardware), and to have just one unit that performs addition on any pair of numbers that is selected from memory.

The two-mechanisms account suggests that the human cognitive architecture utilizes attention to provide a serial mechanism to implement some functions that would be too costly to implement everywhere. Attention, then, is like the access from a computer's memory of items upon which especially powerful processes may run while they are loaded in from memory.

This may explain why so many processes in cognition are not open to introspection — if introspection is a function that is too expensive to "implement" in parallel over the entire brain, then it will take place only in select areas. Another perspective is that self-awareness must necessarily be limited, because if self-awareness is a mental process, then self-awareness of all mental processes would include awareness of self-awareness, and this would lead to an infinite regress. With limited mental resources, some mental functions must be limited in their scope. Attention may be understood as the "promotion" of information from long-term memory to short-term memory, where the specialized operations are supported.

2.7.2 Nomenclature

It is difficult to find terminology for these mechanisms that is accurate and descriptive, but not tainted by unintended connotations. All the most obvious ways to denote them involves terms which have already been appropriated by other authors with different purposes in mind. In addition, there are many ways in which to contrast the two types of mechanism, and the desired point of emphasis may be different from one circumstance to another. With this in mind, the rest of the thesis will primarily use the terms “distributed” and “algorithmic” to describe the two mechanisms, but they could also usefully be described as “preattentive” and “attentive”, or “fast” and “slow”. The underlying mechanisms will, as above, be referred to as “flat” and “structured”, respectively.

“Distributed” calls to mind the connectionist tradition [McClelland and Rumelhart 1986]. In connectionism, the word “distributed” is often applied to the manner in which networks can represent information over a set of weights or activations, often in a non-localized way that makes it impossible to say where in the network any specific item is represented. Here, the term is used to emphasize the idea that in the visual domain, processing is not localized, but rather is distributed, and takes place in parallel over an image map; no claim is made here that the distributed mechanism employs non-localist representations.

The choice of “algorithmic”, likewise, does not imply an exact correspondence between the mechanism described here and the serial execution of a program on a digital computer. It does, though, suggest that certain similarities exist. Both use short-term memories that can build structures, in which symbol-binding acts to instantiate representations that combine multiple features together. Both the human and computer versions operate in serial while making use of their own respective stores of memory.

2.7.3 Comparison with Sloman's two types of reasoning

As Sloman notes, it is difficult to come up with a set of contrasts that exactly and completely distinguish two different modes of cognition such as are addressed in Table 2.1. Many or most of the contrasts fail to hold in certain instances. Properties meant to contrast sometimes co-occur; properties meant to correlate sometimes fail to do so.

It is doubtful that more careful consideration could retool the table so as to be precise, complete, and general across all possible tasks. It may be possible to identify and characterize a contrast between mechanisms when (and only when) one task is specified as the object of inquiry. This subsection attempts to provide whatever level of detail is possible, based on the experiments done thus far, regarding how Sloman's contrasts apply to the two mechanisms seen to be at work in letter categorization.

Source of knowledge

Sloman argues that the kinds of information that his two systems of reasoning utilize come to the person by different means. The experiment reported here does not directly address *how* the knowledge of the alphabet required for each mechanism's approach is acquired (it *is* acquired; knowledge of the alphabet is obviously not innate). An oblique approach could start with the assumption that the use of attention applies only to the algorithmic mechanism, and go on to note the same distinction in a purported dichotomy between explicit and implicit memory, also dependent upon the use of attention [Schacter 1987; McClelland et al. 1995]. One might then guess that the algorithmic mechanism depends upon information in explicit memory, and that the distributed mechanism depends upon information in implicit memory. This claim cannot be made with any certainty at this point, but is a starting point for further exploration.

Nature of representation: Basic units

This contrast of Sloman's corresponds tightly to this chapter's claims regarding the two representation types.

Nature of representation: Relations

Some of the terms in this row of Sloman's table may have different meanings for different readers. That structured representations are hierarchical is clear.

As for hard versus soft constraints, this distinction is not a rigorous one. However, there is some justification for saying that something like that distinction applies to the distributed–algorithmic divide. Category-membership, as decided upon by a flat-representation mechanism, is a matter of the degree of overlap between the instance in question and the category prototype. As a matter of degree, this is consequently a matter of soft constraints. Categorization by a structured-representation mechanism, however, might be very sensitive to the presence or non-presence of a suitable role-filler for each of the roles in some particular category's prototype. Whether or not a whole is divided into parts or not, and each of the expected parts present, *is* a black-and-white matter; in this sense, the algorithmic mechanism can be said to depend upon some hard constraints.

Nature of processing

The evidence presented here for dual mechanisms in letter recognition is based on representations, and does not directly say a great deal about processing, which is a different (but not altogether independent) issue. Indeed, the early papers assumed that flat representations were utilized, and those papers focused almost exclusively on the processes acting on the representations as the variable in their models. Clearly, at least for *hypothesized* models, the representation does not determine the details of processing.

At present, there is little evidence regarding the fine details of process. Sloman's points (b) and (c) (under "Nature of processing") are compatible with what has been said thus far, though point (a) is too vague to agree or disagree with. "Similarity" begs the question of what brand of similarity — each type of representation entails its own intercategory metric, and thus defines its own brand of similarity.

That caveat regarding "similarity" can be extended to other terms, such as "categorization". For example, Lakoff's approach to categorization rejects certain models (such as weighted feature bundles) because they are inadequate as the basis for certain categories [Lakoff 1987]. If, however, there exist multiple mechanisms in the mind, each of which is capable of supporting categorization, then demonstrating that one particular mechanism is inadequate for recognition of a certain, limited set of example categories does not exclude the possibility that the given mechanism could account for categorization involving *other* categories. While Lakoff shows that weighted feature bundles are inadequate for certain complex categories, they may still prove to be a good model for categorization in other areas. In particular, the Examiner uses something very much like weighted feature bundles to categorize parts as role-fillers. It may be that this is a good model of categorization on that level, while categorization in other situations (for instance, on the role level) involves a more elaborate mechanism. The issue is explored further in later chapters.

In general, the possibility of two mechanisms at work on the same task makes some kinds of conclusions based on experiments (or on one experiment) a bit risky. As the conflicting papers on letter categorization show, it is possible to experimentally find the results of one mechanism and to conclude that it is *the* mechanism at work, when in fact it may be one of co-equal alternatives. This calls to mind John Godfrey Saxe's poem about six blind men, each drawing a different conclusion about the nature of an elephant after each has felt a different part of the animal. It has already been well accepted for some mental phenomena that what naively seemed like one facility is actually more than one; no one speaks, any longer, of “memory” without distinguishing short-term memory from long-term memory, for instance. Evidence of dual mechanisms suggests that greater caution should be exercised in concluding that any one experiment has demonstrated *the* way that humans solve a problem.

An approach to dual modes of cognition (clearly trying for a distinction similar to that raised here and by Sloman) that focuses on the nature of processing is found in [Smolensky 1988]. Discussion of processing in the two mechanisms is addressed more later in this chapter.

Illustrative cognitive functions

This portion of Sloman’s table lists processes and domains where he suspects that one type of reasoning, rather than the other, is at work. Future research might explore these cases individually, to probe each one for the exclusivity of one mechanism (as opposed to the influence of multiple mechanisms), and if more than one mechanism is found to be at work, then to probe for the nature of those mechanisms and of their interaction.

One item that deserves exploration in much greater detail is “creativity”, which will be dealt with at length in Chapter 4.

2.7.4 Other contrasts

Beyond those contrasts that Sloman proposes for his two systems of reasoning, there are many others that one might use to characterize the two mechanisms involved in letter categorization. One might assume, from contrasts such as those in “Illustrative cognitive functions”, that the two modes of cognition effect a division of labor, and that they do not act in the same domains.[2] Obviously, that is not the position taken here regarding letter categorization. A number of important contrasts between the two mechanisms suggest how they might contend for influence or cooperate while participating in the same task.

Accuracy

It seems likely that the distributed mechanism is intrinsically less accurate than the algorithmic mechanism. If the behavior of the distributed and algorithmic mechanisms can be considered to have been isolated in the fastest and slowest conditions, respectively, then one could conclude that the distributed mechanism gives answers that are correct about 42% of the time on this data set, while the algorithmic mechanism operates at about 75% accuracy. This simple approach, while possibly sound, is beset by complications. One problem is that a given experimental condition may not demonstrate the work of one mechanism in complete isolation of the other. Also, the distributed mechanism may suffer degraded performance when it is given so little time to operate; perhaps it generates better answers, by means of the same mechanism, when allowed longer access to the stimulus.

There is, however, other support for the superiority of the algorithmic mechanism. There is evidence that decompositional approaches (such as the algorithmic mechanism) are superior to other approaches in the recognition of gridletters [McGraw 1995]. Finally, one is left with the fact that the algorithmic mechanism has the larger influence when both have sufficient time to work. If the distributed mechanism did, in fact, lead to better answers than the algorithmic mechanism, and yet if they

[2]Sloman does not make this claim [Sloman 1996]; he makes clear that he finds the opposite to be true, but does not survey in detail in which domains and in which ways he believes the two modes may both act.

were for some reason discarded, that would leave the puzzle as to why such a counterproductive strategy would be employed. All in all, it seems likely, if not certain, that the algorithmic mechanism leads to generally more accurate answers than the distributed mechanism.[3]

If level of accuracy does distinguish the models, then it suggests that there must be some advantage in the distributed mechanism's favor, or there would be no purpose in its existing at all. Speed is one probable advantage. Other possible reasons for the existence of distributed perceptual mechanisms will be discussed in Chapter 3.

Process models

The approach so far has focused on representation, not process. Even so, it is not hard to see why one might conclude that the algorithmic mechanism resembles rule-based reasoning and symbolic artificial intelligence, while the distributed mechanism is akin to connectionist models. In the first case, consider the likely specifications for a letter recognizer that treats letters as structured entities consisting of multiple roles. A straightforward approach would call for embedded, hierarchical representations, for the ability to dynamically create representations for parts as they are identified in the letter, and for the ability to bind a given part with one or more roles. This short list reads like a roll call of those ways in which classical symbolic AI notoriously has an edge over standard connectionism [Smolensky 1988].

Flat representations require none of these things, and reduce the problem to the sort of pattern-matching problem that connectionism has been shown to excel at. (Smolensky argues along these lines.) Additionally, a table similar to Sloman's, but

[3]This may not be true in all tasks. In both professional basketball and professional football, when a player is allowed one last potential game-winning shot or kick, the *other* team routinely calls a timeout, in the belief that having more time to think about the importance of the play will make the player more likely to miss. The deleterious effect of the timeout on the player's chances may be a fact, or else it may simply be a very persistent myth.

couched precisely in terms of cognitive models (and with a third column describing "dynamical" processing — not necessarily akin to either of the two mechanisms as described here) is found in [Van Gelder 1997].

A simple way to explore the distributed–connectionist similarity is to compare the experimental and theoretical matrices with the confusion matrix generated by a three-layer connectionist network trained by backpropagation for the gridletter recognition task ("Netrec") [McGraw 1995]. Table 2.6 shows that the error matrix for the simple neural network correlated quite well with the flat-representation matrix, but did not, surprisingly, correlate any better with human data in the 17 ms condition than in the 117 ms condition. This probably suggests that both of the theoretical models fail to capture the corresponding human mechanisms very precisely.

17 ms	*117 ms*	*Flat*	*Structured*
0.332	0.322	0.506	0.322

Table 2.6: Pearson correlations between the performance of a three-layer backpropagation network and experimental conditions and theoretical models.

The claim that the two mechanisms (algorithmic and distributed) correspond to the two camps in AI (symbolic and connectionist, respectively) can only be as strong as the broad generalizations of symbolic AI and connectionism upon which it is based. It may not be accurate to say that the division of artificial intelligence into symbolic and connectionist camps exactly mirrors the two perceptual mechanisms, but it is still interesting that two such mechanisms exist in the mind, and that rough correlates of them surfaced in the first few decades' worth of efforts to model the mind computationally.

Memory/Abstraction

The investigation thus far has not directly taken up the issue of memory. However, the contrasts in Sloman's table mirror, in many ways, those proposed for explicit and implicit systems of memory [Schacter 1987; McClelland et al. 1995].

A finding related to the mechanisms as described so far comes from neuroscience. [Desimone et al. 1995] identify three types of short-term memory in visual cortex. One is automatic and involuntary (it acts even in anesthetized animals), operates very quickly, and seems to reside entirely within inferotemporal ("IT") cortex. By means of this mechanism, IT cells that fired due to presentation of a stimulus in the recent past ("matching cells") have a relative advantage in subsequent presentations of matching stimuli, even if the match is not an exact one, thus demonstrating generalization. The second kind of memory involves a mechanism by which pathways from prefrontal cortex to IT enhance activation in matching cells that correspond to the contents of working memory. (The third kind is addressed a bit later.)

While memory is hardly the same thing as reasoning, one should note that the contrast between the first two kinds of visual memory listed above is strongly reminiscent of that between Sloman's two systems of reasoning. One system is fast, involuntary, and preattentive, while the other is slower, using working memory as a store, which allows for abstraction, symbol-binding, and the building-up of arbitrary structure. The two contrasts may illuminate one another; for instance, the neurology of the two perceptual mechanisms may be related to or overlapping with that of the corresponding memory mechanisms.

The third kind of memory for which Desimone and colleagues find evidence is one in which activation of matching cells (those that have matched a stimlus presented in the recent past) is sustained for a short time, in the absence of any new stimuli. This does not directly match either of the two mechanisms suggested so far, but does suggest a kind of memory consisting of activation (or lack thereof) of the entire set of concepts represented in IT cortex. This has some properties that suggest localist PDP models, as well as the Conceptual Network used in Letter Spirit (see Chapter 5). The relationship between the dual mechanisms underlying categorization, on the one hand, and implicit and explicit memory, on the other, merits future exploration.

Figure 2.4 summarizes many of the properties mentioned so far for the distributed and algorithmic mechanisms. More detail in table form follows in Table 3.4.

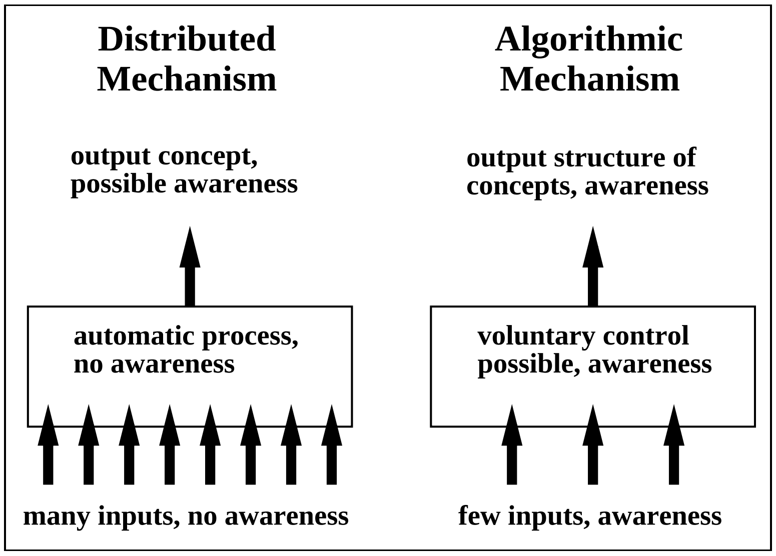
Distributed Mechanism
Algorithmic Mechanism
output concept, possible awareness
output structure of concepts, awareness
automatic process, no awareness
voluntary control possible, awareness
many inputs, no awareness
few inputs, awareness

Figure 2.4: A number of the key characteristics distinguishing the two types of mechanism.

2.8 How the two mechanisms may relate

2.8.1 Why two mechanisms?

The first question to ask about a dual-mechanism system of perception is: Why have both mechanisms? One answer is that it allows the organism to take advantage of whichever mechanism is better suited in any particular situation.

The advantage of the dual mechanism can be illustrated by means of a hypothetical scenario. Suppose that the distributed mechanism, for a certain task (perhaps telling the difference between predator and prey in the jungle) calculates answers instantly, and is correct 45% of the time. In contrast, suppose that the algorithmic mechanism takes 75 ms to act, but is correct 90% of the time. Consider three beasts in this jungle.

One has only the distributed mechanism at its disposal, one has only the algorithmic, and the third beast has a hybrid perceptual system employing both mechanisms, and elects the better answer (the algorithmic) whenever it is available. A simplified environment that leads to a large number of encounters in the jungle that allow 50 ms for the correct identification to be made, and an equal number of encounters that allow 100 ms, yields the performances in Table 2.7:

	50ms events	*100ms events*	*All events*
Distributed only	45%	45%	45%
Structured only	0%	90%	45%
Both systems	45%	90%	67.5%

Table 2.7: The best of both worlds: A dual perceptual system leads to greater accuracy than either of its component systems could achieve alone.

As is often seen in computational problems, the best solution is to have two approaches, playing both sides of a tradeoff, and to employ the one that works better in the given situation. The memory hierarchy in digital computers is one example: many computer systems have several types of memory (CPU, cache, RAM, hard disk) in which capacity and speed relate inversely. The operating system keeps data in the fastest type of memory big enough to hold it, and thus maximizes speed and capacity in a way that would be impossible with only one type of memory.

The same phenomenon can be seen in countless non-computational situations — transportation, for example. One walks to the corner store, drives across town or to a nearby city, and flies across the continent. Again, the hybrid solution is vastly superior to what would be available if one had to choose only one of the available options. The mouths of omnivores (such as people) contain flat teeth for chewing as well as sharp teeth for cutting. A parrot's beak provides a similar range in surfaces, all in one structure. The Swiss Army knife and the spork (spoon–fork) show the toolmaker's rediscovery of this principle: If having one function is good, then having two functions is better. Similar arguments for the usefulness of complementary systems have been made by [Smolensky 1988], [Boden 1990] and others.

2.8.2 Interaction

Parallel Mechanisms

Up to this point, the only function that has been proposed for the two mechanisms is that they both offer answers to a categorization problem, between which a person can somehow choose. This account is simple enough; however, there is no solid evidence that it is accurate, much less complete. There are a number of possible ways in which the mechanisms could be utilized by the perceptual system.

One possibility is that one mechanism, or both, or the perceptual system that makes use of them, could have the ability to revise their answers as time passes, presumably for the purpose of improving accuracy. However, it appears that very little changes for stimulus durations over a few hundred milliseconds. Table 2.8 shows the accuracy and correlations garnered from all durations probed in all three versions of the experiment, demonstrating that the trends established over shorter durations appear to slow and then halt as durations grow longer. Apparently there is a ceiling on performance, beyond which additional time does not alter performance (either accuracy or the correlations) very much. The way that performance changes, or ceases to change, as additional time resources become available to the subject is obviously task-specific. [Goldstone et al. 1991] report a different categorization task that shows a gradient in the nature of behavior as a function of time. In that study, the intervals over which performance was found to vary were longer than those in the letter-categorization experiment, perhaps because the stimuli were more complicated.

A second unknown in the letter-categorization paradigm is how the decision between the two mechanisms is made. It may be that the algorithmic mechanism's output is preferred whenever it is available, or it may be that some test (voluntary

Stimulus duration	*Accuracy*	*Flat correlation*	*Structured correlation*
17 ms	41.8%	0.597	0.509
33 ms	52.1%	0.605	0.606
50 ms	64.8%	0.527	0.615
67 ms	67.1%	0.496	0.716
100 ms	75.1%	0.501	0.735
117 ms	71.8%	0.535	0.686
400 ms	78.4%	0.490	0.716
800 ms	76.6%	0.431	0.723

Table 2.8: Correlation with models vs. stimulus duration.

or otherwise), distinct from either mechanism, is applied to determine which output will be used. The data in Table 2.8 suggest that context may play a role. The way that the influence of one model drops off as the other picks up is nearly monotonic over stimulus duration, all the way from 17 ms to 800 ms. An exception is in the 100 ms and 117 ms trials, which show a strange spike in the opposite direction. Given the similarity of the two durations, one would expect little difference between these conditions, much less in this direction. One possible explanation is that the behavior in each condition was influenced by the other durations with which it was grouped. 100 ms was part of an experiment with 400 ms and 800 ms trials, while 117 ms was part of a set with 17 ms and 50 ms trials. It may be that subjects came to rely on the algorithmic mechanism if all of their trials permitted this, while subjects who had two-thirds of their trials so fast that only the distributed mechanism's output was available began to rely on it. Thus, the 117 ms trials reflected a greater influence from flat representations. A followup would be needed to make sure that this effect is not merely noise in the data. For now, all that it is safe to say is that two processes take place, that subjects have one or two outputs to choose from, and that a choice is made at some point.

Interconnected Mechanisms

A more complex hypothesis, not original to this thesis, is that the output of a fast mechanism may serve as the input (or part of the input) to a slower mechanism. Several models describe visual perception as a process that builds up from one level to another in a chain of successive processing steps [Marr 1982; Ullman 1989; Biederman 1987]. In these models, lower levels of representation are the building blocks for higher levels, which are calculated subsequently and are derived from those lower levels. Those models call for something like Account 1 in Figure 2.5. The results of this chapter suggest that Account 2 is more apt — that even the early levels can result in a high-level output, such as categorization, without the need for an intermediate step. Thus, the perceptual system is opportunistic, allowing the system to reap benefits like those displayed in Table 2.7.

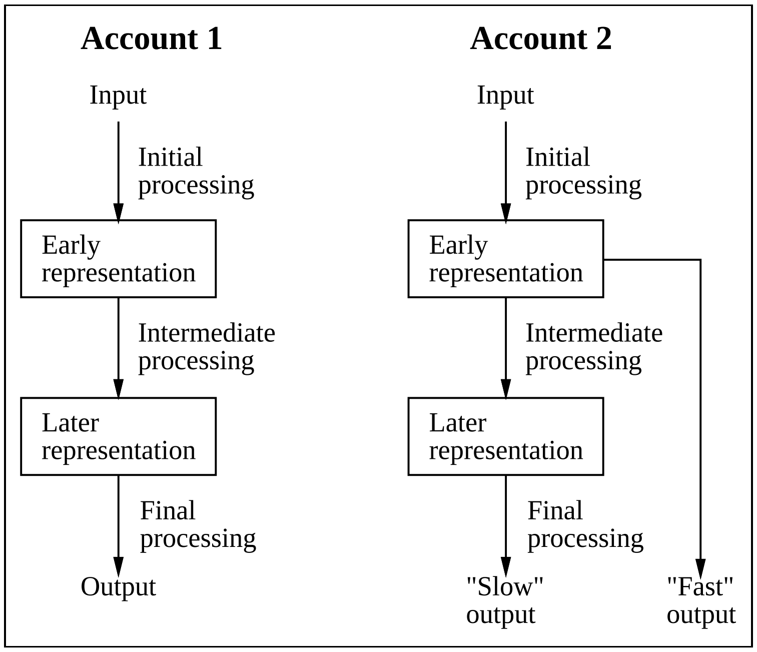

Figure 2.5: Straight-line processing provides a simple model.

Intertwined Mechanisms

Diagrams such as Figure 2.5 do not capture time well, and make it easy for the reader to overlook the fact that the faster output of the distributed mechanism is already calculated while the algorithmic mechanism that generates the slower output may just be getting started.

If the system as described thus far were (hypothetically) a computer program whose completion were up to a team of human programmers, they would be apt to use the distributed mechanism as a heuristic to guide the algorithmic mechanism. After all, the distributed mechanism meets the most important requirements of a heuristic: that it operates quickly and that it gives an answer that has a good chance of being correct.

Turvey provides some evidence that central processes depend upon the output of peripheral processes. Moreover, as other work has shown, there is reason to believe that it could be beneficial to overall performance to utilize it in that way. The interactive-activation model demonstrates this, actually using stimuli that are similar to gridletters — letters composed of segments, in small arrays capable of spelling short words [McClelland and Rumelhart 1981]. The model attempts to resolve what word is represented, though the display has some noise in it (not all the quanta that should be turned on to spell the word are actually turned on). The finding was that the processing that works from the bottom up is assisted in resolving noisy data on the lower levels (letter and components of letters) by top-down pressure from the word level. The processing that attempts to activate the correct word node based upon the input benefits from top-down information at the word level. In the case of the interactive-activation model, the activation on the higher level happens to originate from the lower levels (working with imperfect information). It seems reasonable to assume that good "hints" from the higher levels would have value no matter what their origin, and in the case of the letter-categorization paradigm discussed here, the top-down hints for the algorithmic mechanism could come from the distributed mechanism.

In light of this analysis, two additional accounts for dual-mechanism perception are given in Figure 2.6. On the left is a simple straight-line model that utilizes top-down influence to guide future processing. On the right is the most sophisticated model offered in this chapter — one that utilizes two parallel mechanisms and also allows top-down influence. Because one mechanism is faster than the other, it has a preferential role in providing top-down influence for the other.

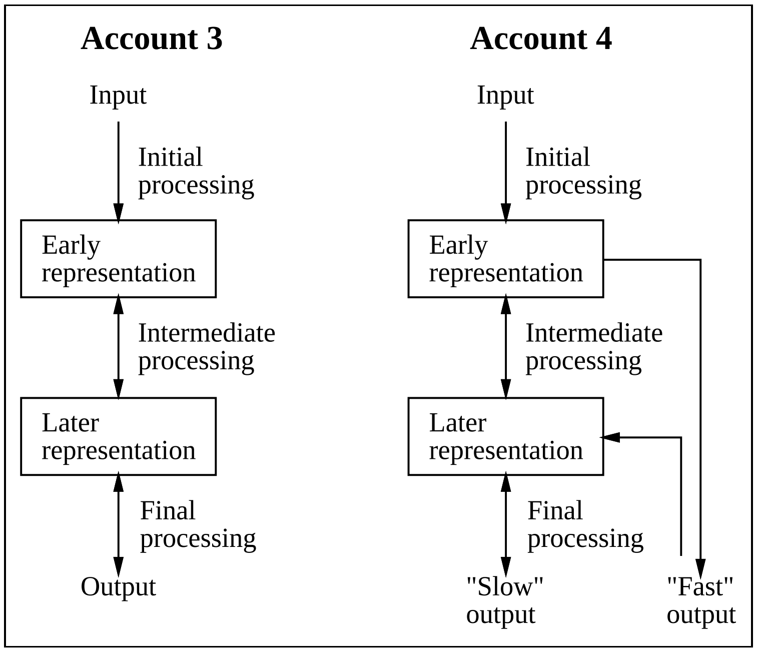

Figure 2.6: Top-down influence is a part of more sophisticated — and more complete — models.

Up to this point, it has been implied that the two mechanisms each offer one answer at a time. However, if the parallel nature of the distributed mechanism extends to the output as well as the input, then it may serve to activate a large number of possible answers at the same time; in the terminology of functions, this is high fan-out. If the distributed mechanism is parallel in the extreme, it may activate each possible answer to an extent that is proportional to how well that answer is supported by the input. This is the paradigm in countless connectionist models of cognition. If it is an accurate depiction of the distributed mechanism, then the algorithmic mechanism could hone its performance by paying more attention to the categories rated most highly by the distributed mechanism.

A computational experiment based on computational models of letter recognition can show the benefits of employing a strategy like Account 4. Table 2.9 shows the performance of three variations on a model (Letter Spirit's Examiner) to support the point. Here, three versions of the Examiner are used to demonstrate the power resulting from the use of a fast, distributed-processing letter recognizer as a heuristic guiding the algorithmic behavior of a structured-representation model. The first model is a crude but quick "gestalt" function that attempts to categorize the gridletter using holistic properties based on overlap with the category prototype (the details of the Examiner's implementation can be found in Chapter 5). The second model uses a more algorithmic style of processing based on structural decomposition, like the structured-representation mechanism described above. The third model is exactly like the second, but begins by running the gestalt recognizer as a module that computes a score for each of the 26 categories; this score is utilized in a number of ways by the rest of the model. All three versions were run on a data set of 415 gridletters, to compare their accuracy and mean speed, time being measured in units of processing called codelets.[4]

In terms of the hypothetical jungle, the potential advantage enjoyed by the beast with dual mechanisms for visual categorization is even more apparent than before. The two mechanisms support each other by acting in complementary circumstances,

[4]The gestalt function's operation is not divided into codelets, so no times are given for this. It suffices to say that the actual running time for the gestalt function is much faster than those of the structured representation models.

	Percent Correct	*Time*
Gestalt only	57.3%	does not apply
Structured only	85.3%	863.3
Structured with gestalt	91.6%	508.8

Table 2.9: The Examiner profits from using both types of mechanism in an intertwined fashion.

and by acting collaboratively in other circumstances, with the faster as a heuristic for the slower. It remains to be seen if visual processing in humans does in fact make use of one mechanism as a heuristic for the other, but the principle is potentially very useful. It may not be possible to create a circumstance where human subjects are examined first with, and then without, the use of the distributed facility, to see if it does have a role in the behavior of the slower mechanism, but evidence for such a top-down effect may come from analysis of residuals in long- exposure trials, and perhaps from neurological studies.

The use of heuristics is, of course, widespread in AI. An application very similar to what is proposed here, and implemented in the Examiner is the handwriting-recognition unit of the Apple Newton handheld computing device. That system uses a connectionist approach as a heuristic for an algorithmic approach that makes the final decisions regarding word and letter category [Larry Yaeger, personal communication]. That the Examiner and the Newton should both employ that basic strategy suggests that there is a convergence between accurate models of human letter recognition and what happens to work well.

2.9 Conclusion

Although the basic idea of two kinds of cognition is not original, the degree of support that has been made possible by focusing on a single task appears to be unprecedented. The framework established in this chapter is crucial to the rest of the thesis. The next two chapters depend upon it, and much of the rest of the thesis depends upon them. Those topics, in turn, are essential to gauging what the contribution of Letter Spirit is to the understanding of human abilities in aesthetic perception and aesthetic creation.

In addition, the framework of activity within one domain, but based on two kinds of cognition, seems to have generality beyond the domain of letters. When this work was in an earlier stage, it was discovered that work in studies of language comprehension also explicitly postulated two kinds of cognition at work [Christianson et al. 2000]. Previous FARG models, as noted earlier, entail such a distinction, though without putting such a fine point on the kind of interaction between the two kinds of cognition. Models by many other researchers, in domains ranging from categorization by olfaction to analogy-making [Forbus et al. 1995] employ dual mechanisms like those proposed here. It seems odd that, despite the mounting evidence for the existence of two such mechanisms, so many models of cognition are put forth without the explicit specification of whether they model the activity of distributed mechanisms, algorithmic mechanisms, or both.

CHAPTER THREE

Quality and Style

3.1 Introduction

De gustibus non disputandum est.

(There's no accounting for taste.)

— Unknown

The field of aesthetics has long grappled with (among other concerns) two related questions: "What is beauty?" and "What is style?". Although the questions are not precisely the same, in the context of typefaces the concerns are similar, because in typeface design the goal is to achieve consistency in style; therefore, beauty is found where style is felt to be constant across many letters.[1] One may make the very general observations that beauty entails pleasure and that pleasure entails motivation (the motivation to view or pursue the beautiful object), but the real challenge is to

[1]Avant-garde typefaces may shun *surface* homogeneity, and even look wildly inconsistent in style [Quart 1999]. Some published typefaces even have multiple tokens for the same character, and the choice of which to use is made by context, or even randomly [Meggs 1998]. This, however, is simply a type of consistency on a more abstract level. Gridfonts exploring this sort of "radical" consistency were designed soon after the gridfont domain was created [Hofstadter 1985].

identify *why* an object elicits this response. In general, the question of beauty and the question of style seem comparable in complexity, and both of them concern what properties of an image evoke certain aesthetic reactions in a viewer. They were first grappled with well over two thousand years ago [Plato 1974; Aristotle 1963], but in spite of this, completely satisfactory answers have never been found. There is the ubiquitous observation that although individuals can identify what things they find aesthetically pleasing, they cannot say why they find those things pleasing. Even as of very recently, semioticians have said "it is difficult, if not impossible, to give a semiotic definition of style" and that style "has never been given a semiotic definition" [Nöth 1995]. Attempts to define beauty, style, and quality frequently conclude that one of the more viable strategies is to quit! Sheppard [1987, p. 61] considers this one of the more (but not the most) tenable stances:

> ...we might answer the request for a definition of beauty by declaring that it is a simple quality which cannot be further defined in terms of anything else. We recognize it by intuition and there is little more to be said.

Pirsig's personal quest for a definition of quality [Pirsig 1974] leads him eventually to declare that term (as he uses it, essentially a synonym for "beauty") undefinable.[2] Machotka [1995] reluctantly finds the most obvious approaches to aesthetics fruitless and declares that "the question for us now is to what degree, or in what sense, aesthetics is possible at all."

It certainly must be a difficult question that drives researchers to declare it unanswerable! However, it is a question that cannot be sidestepped in work on a project

[2] "Quality" is used by various writers in at least two senses. One is a noun indicating any characteristic of an object. The second is also a noun, but indicates that an object is highly valued with regard to some way of considering the object. The second sense is, in discussions of aesthetics, equivalent to "beauty". The second is distinct from the first, and it should always be clear from the context which meaning is intended.

like Letter Spirit. For a computer program to be able to review and revise its own work, it must be able to evaluate the degree of aesthetic excellence of that work. This involves assessing the *beauty* of the work, or the degree to which the work fits the intended *style* (hence the equivalence of the two questions, with regard to typefaces). However the problem is phrased, Letter Spirit demands a solution — at least a partial solution — to one version of a very difficult and ancient problem. A rough account of how Letter Spirit handles style appeared in earlier writings while the project was still in early development [Hofstadter and FARG 1995]. The real contribution of Letter Spirit to the study of aesthetic perception will be not only that the program works, but also the light that is thereby cast on the more general topic of visual beauty. By employing a definition of style that works in the gridfont domain, Letter Spirit has made some headway towards an answer to the general question of beauty and style, although a definitive and comprehensive answer for the general case lies, to be sure, far beyond the range of this project.

3.2 Subjectivity in aesthetic responses

Beauty is in the eye of the beholder.

— Margaret Wolfe Hungerford

One complication is that the perception of beauty, or quality, or style, is without a doubt subjective. To prove the point, one need only find a modest corpus of reactions, by different people, to the same works of art. Comparing these reactions is often complicated by the fact that quantitative ratings of works of art are anathema to most art critics. In artificial intelligence and cognitive science, by contrast, goodness ratings are rampant. Game-playing programs use static-evaluation functions [Samuels 1967], and psychologists ask subjects to come up with quantitative ratings expressing their approval of an object or idea [Fechner 1997]. The reluctance that art critics have towards quantitative assessment is worth pondering; some objections will come up in the discussion of "aesthetics from below" in the following sections. For now, it is enough to accept the good fortune that film critics *do* issue quantitative ratings with their reviews. Ratings by two different respected film critics of the films with the highest box-office gross in the United States in 1998 provide a point of comparison. Table 3.1 shows how the five most popular movies of 1998 were rated on the common system of zero to four "stars" [Ebert 2000; Maltin 2000].

	Ebert	*Maltin*
Saving Private Ryan	4	3.5
Armageddon	1	2
Something About Mary	3	2
A Bug's Life	3.5	3.5
The Waterboy	1	3

Table 3.1: Two critics' ratings of the same movies.

A third judgment of taste is also involved, in that these movies were all ranked highly according to ticket-buying preferences of the American moviegoing public. If aesthetic taste were objective, the two critics would rate these movies similarly, and to conform to audience opinion, these ratings would have to be quite high. Obviously, no such conformity exists. The correlation between Ebert and Maltin's ratings through the twenty-five highest-grossing movies of 1998 is modest ($r = 0.322$, $p = 0.116$), and, even if the two critics do have some similarity in their tastes, they also clearly have their differences. Something that highlights this is the wide discrepancy between their ratings of *The Waterboy*. The ratings are accompanied by reviews, which fail to explain the discrepancy. Ebert identifies the central character as "insufferable" and "annoying". Maltin, however, uses terms nearly opposite in meaning, calling the same movie a "likable, crowd-pleasing comedy with heart." The success of the movie suggests that a large number of people agreed with Maltin and disagreed with Ebert. The question of *why* reactions can differ so greatly is deferred until later in this chapter, but it is useful to have established the fact of subjectivity now.[3]

3.3 The dilemma in studying style

Your approach to painting seems to be torn between two opposing tendencies: the cold observation of detail, and the flaming abundance of life. Are you like Dürer, or are you like Titian? Are you a German, or an Italian? To want to be both is a great ambition. But you haven't yet resolved the conflicts.

— Honoré de Balzac, *The Unknown Masterpiece*

Although some have given up on the question of beauty, others have offered definitions. As might be expected from the pessimism of the "quitters", no definition has ever seemed satisfactory to all. This section offers an account of the two dominant camps among those who try to define "beauty". The division between the two camps corresponds roughly to those who take a holistic approach ("from above") and those who take a reductionist approach ("from below"). Each side of the issue has its detractors, especially among those on the other side.

3.3.1 Holism: The approach "From above"

The largest body of commentary on art comes from art historians and critics. A holistic approach to defining beauty, as described by Sheppard, comes logically after creation and evaluation of art, and consists of "consideration of what sort of judgment we make in judging something to be beautiful." Sheppard holds that analysis of commentary on art, particularly evaluative and comparative statements made regarding

[3]An additional perspective on subjectivity is found in [Kant 1977], pages 647–648. Kant argues that when a viewer declares an object beautiful, speaking as though the beauty were a quality of the object, the viewer must believe that the object's beauty is universal, and not dependent upon the viewer. This seems plainly false. The observation, post-Kant, that the semantics of an utterance need not be reflected in its surface structure supports the possibility that such statements are shorthand for expressing an opinion.

the quality of objects of art, is a way to find the meaning of beauty — in fact, the only way that looks "profitable" [Sheppard, p. 64].

The recurring criticism regarding this approach is that the commentary of art historians and critics is unscientific, fuzzy, and unpredictive. In the view of Vitz and Glimcher, the holistic approach is plied by those who fail to understand how helpful the tools of science could be for aesthetics. They also feel that in art history and criticism, "One often finds vocabulary so obscure and ornate, and syntax so convoluted that the writer's meaning is lost in the flow of words. Indeed, on occasion one wonders what idea, if any, the writer had in mind" [Vitz and Glimcher, 1984, p. 9]. Boselie and Leeuwenburg [1984, p. 367] find two flaws in the theories of art historians and critics. One is that they advance, as though meaningful, vacuous hypotheses that both of various pairs of opposing qualities (such as unity and diversity, or order and complexity) are necessary for excellence in art. A broader complaint is that none of their theories tend to be stated in a formal way, and therefore cannot be falsified. Machotka [1995] finds holistic aesthetics ("aesthetics from above") simply "undesirable".

The comments just quoted are not altogether fair, in that art critics and art historians are not necessarily *trying* to explain art in a way that reduces to a formal system or anything grounded in simple axioms. They write for readers who have working, sophisticated visual systems and some knowledge of the pragmatics governing the subject matter that art depicts. Those who write about art are not compelled to derive a theory so thorough and assuming so little that not only could a child understand it, but that it could easily be turned into a computer program that evaluates art. Consequently, criticism of aesthetics from above is not tantamount to a declaration of war upon the vast body of art critics and art historians. However, the crucial point is that those who *have* attempted to deliver a comprehensive theory of art grounded in "data" derived from opinions and commentaries by art critics and art historians have not been able to do so.

3.3.2 Reductionism: The approach "From below"

Given the way that intellectual history has gradually turned towards a more scientific orientation, it is somewhat surprising that reductionist approaches to art go back to the earliest records. Around 450 B.C., the Greek sculptor Phidias believed in the harmony of the golden mean, hypothesizing that the rectangle most pleasing to the human eye had sides the ratio of whose lengths was $\frac{1+\sqrt{5}}{2} \simeq 1.618$. This hypothesis was explored experimentally by Fechner in the nineteenth century [Fechner 1997], and has continued to inspire research even into the 1990s [Barratt 1994]!

The approach from below appears to be flawed, though. Fechner apparently recognized the arduous path that his "from-below" approach entails, as he called his work a *Vorschule* — the preliminary to a study. Indeed, all of the theories based on the approach from below are incomplete; none of the theories regarding the Golden Mean, the structures described (or prescribed?) in Aristotle's *Poetics* and later by Formalists and Structuralists in literary theory, nor the work of Birkhoff [1933] constitute a complete theory of what determines quality in a work of art. It may be that continued application of the approach pushes the level of understanding subtly forward [Barratt 1994]. The slow pace of progress, however, has undoubtedly been one source of frustration regarding reductionist approaches to aesthetics. Sheppard's rejection of aesthetics from below is empirical, based on the observation that all prior attempts to specify in this manner what things are beautiful go awry.

> "Attempts to define beauty in terms of particular non-aesthetic qualities are always open to counter-examples; suggested definitions are always too narrow, in failing to include instances of beauty, and too wide, in failing to exclude instances which have the relevant non-aesthetic qualities and yet are not beautiful." [Sheppard, p. 63]

Beauty, of course, is not the only concept that is hard to define. Wittgenstein cites the difficulty of finding a set of features sufficient features for membership in a category, and offers many examples [Wittgenstein 1953].

Machotka [1995] also cites the empirical fact that approaches from below have led to meager success, but provides a much more detailed account than Sheppard as to why this might be. He identifies a number of problems that threaten to inherently cripple the from-below experimental approach. First, Machotka writes, it is difficult to obtain clear empirical results regarding aesthetic evaluation. Science seeks to control variables and to examine factors in isolation; this seems impossible in the study of art. For example, determining the ideal ratio of the dimensions of a painting requires data on viewers' reactions to the painting, which also depend upon the subject matter of the painting. The independent factors in aesthetic evaluation seem far more numerous, harder to gauge, and more prone to nonlinear interaction than is seen anywhere in the hard sciences.

Second, Machotka points out that it is hard to know even how to begin an approach from below by identifying the elements that the theory should focus on. For example, the tradition has identified the shape of rectangles as an important issue. If this and other starting points for a program of inquiry are poorly chosen, the research could yield limited dividends. Given that a problem first identified twenty-five hundred years ago is still not fully resolved, one might conclude that poor choices in overall strategy might waylay a from-below approach for a long time before the error is detected and corrected.

Third is the observation that the use of from-below style rules has been on the wane in the creation of new art. Art instruction was once prescriptive with regard to a number of properties that are easy for from-below approaches to focus on — for example, methods of brush stroke, and shapes of rectangular frames. The tendency to prescribe such properties has decreased markedly over the history of art. The notion that good art is that which comes from precise adherence to prescribed techniques and subject matter has receded, in the collective opinion of the artistic community, from the level of a general truth to that of a forgotten absurdity. Great art, in the current conception, can involve any of a tremendous variety of techniques, media, and subject matter. Consequently, the very qualities and quantities that from-below approaches have up to now sought to investigate seem to be categorically disqualified as the bases for a definition of excellence in art. In Machotka's view, this means that the from-below approach seems doomed not only to irrelevance but, because of the trend along these lines, to *increasing* irrelevance.

Machotka's fourth and final concern for the approach is a general observation that might explain, a priori, the third problem — namely, to whatever extent universal properties determining excellence can be identified, artists will seek to avoid those universals. In the thirteenth century, Sienese painters were required to render the madonna in tightly constrained ways. This led to refinement of the precise techniques called for in such paintings, but this exercise gradually ceased to be art and became more of a craft demanding no creativity on the part of the artist. Artists and viewers alike are repelled by such situations. Thus, any from-below approach seems doomed to failure, because if it did identify anything as an undeniable *sine qua non* of excellence in art, and if all artists sought to incorporate that element in their art, it would consequently become something that creative work would start to avoid. Such trends have occurred in the history of music, as constraints were abandoned once composing within them had been sufficiently explored by composers [Boden 1990]. This suggests that at best, a from-below approach to beauty would set art in motion fleeing away from it, and thus would never capture art.

Machotka and Sheppard do not attack from-below approaches in a mean-spirited manner, nor are they cheered by the failures of those approaches. This is not always true of criticism of from-below approaches by from-above thinkers. As has been stressed earlier, the approach from above places great importance on the affective aspects of art while the approach from below often focuses on the dry analysis of the structure of works of art; this difference between the two camps extends to the kind of rhetoric each side employs in arguing against the other. As though to embrace the stereotype of "art world" thinkers that scientists are apt to have, the from-above critics can be emotional, vague, and almost willfully irrational at times in their criticism of highly analytical approaches. In many cases, the criticism can be quite passionate and derogatory. Included below are two accounts from more popular media (the latter being a mainstream Hollywood movie) demonstrating the passionate rejection, and even revulsion, that attempts to explain beauty by reductionist methods induce in some people.

In a personal essay, Pirsig makes the following remarks regarding formal aesthetics:

> These estheticians think their subject is some kind of peppermint bonbon they're entitled to smack their fat lips on; something to be devoured; something to be intellectually knifed, forked and spooned up bit by bit with appropriate delicate remarks and I'm ready to throw up. What they smack their lips on is the putrescence of something they long ago killed. [Pirsig 1974]

In the screenplay for the popular movie *Dead Poets' Society*, a teacher in the classroom reacts as follows to a textbook lesson by a fictional "J. Evans Pritchard, Ph.D" that evaluates poetry in a formal and quantitative manner:

> Excrement. We're not laying pipe. We're talking about poetry. How can you describe poetry like American Bandstand? 'I like Byron; I gave him a 42, but I can't dance to him.' Now I want you to rip out that page. Go on. Rip out the entire page. You heard me. Rip it out. Rip it out! Go on. Rip it out. [...] Rip it out. Rip! Be gone, J. Evans Pritchard, Ph.D. Rip, shred, tear, rip it out! I want to hear nothing but ripping of Mr. Pritchard. We'll perforate it, put it on a roll. [...] Armies of academics

> going forward, measuring poetry. No! I'll not have that here. You're going to learn to think for yourselves, to savor words. [screenplay: Tom Schulman]

In a more scholarly setting, Arnheim is quoted as saying that empirical aesthetics reduces the aesthetic object to the level of ice cream [Höge 1995].

These examples show a number of ways in which the denunciation of aesthetics-from-below becomes passionate. The approach's activity is likened to unromantic, menial, and perhaps violent behaviors. (In the passages above, it is likened to plumbing, marching, and the operation of cutlery.) Its practitioners are compared to amateurs with base standards (teenagers on the American Bandstand television show). The aesthetic object, for the empirical aesthetician, is likened to things invoking only superficial pleasures (sweets, candy, and pop music) the enjoyment of which does not require abstract thought. Violence is suggested as an appropriate reaction to empirical aestheticians. Expulsive acts involving both ends of the digestive tract are alluded to. These anti-aesthetic rants explicitly say that art concerns emotions, but there is also a clear message in how their authors choose to express themselves. These polemics are made more of insults than of logic, and not only argue for an emphasis on emotion in the analysis of art, but also display such an emphasis in the discussion of the analysis of art. This is all a metonymy for the biggest difference between the two camps: feeling versus knowing.

3.4 The aesthetic conceptual hierarchy

Figure 3.1 illustrates the state of research into the nature of beauty. A framework implicit in the terminology of the previous section is represented graphically as the vertical orientation of the figure. That is (as the phrases "from below" and "from

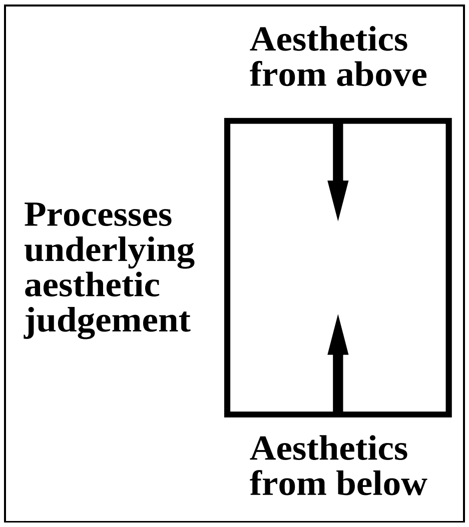

Figure 3.1: Approaches from above and below fail to meet in the middle.

above" imply), the observations of art critics and art historians are based on high-level concepts, potentially quite abstract. The bottom of the diagram represents the low level where aesthetics-from-below tends to conduct its investigations. The belief that the former types of concepts can *potentially* be decomposed into the latter drives the terms "holism" and "reductionism", which have also been used to denote the two camps.

The deadlock here stems from the fact that the two approaches fail to meet in the middle. A thorough understanding of artistic quality would come from an account that penetrated all the way through, explaining high-level observations in terms of low-level properties of works of art. Sheppard notes that attempts to decompose the abstract qualities of art that critics believe lead to judgments of quality do reach downward along the continuum, but fail to hit bottom. Meanwhile, the history of aesthetics from below has barely reached upwards at all. The most fundamental features, such as the ratios of rectangle sides, certainly do not seem to get at anything like the essence of beauty. Aesthetics from below might aspire to work up the ladder of abstraction, rung by rung, until reaching a level where features that really do determine beauty are found. One might imagine levels of abstraction where each level's features are a bit more complex than those in the level below, and are composable from them. Intuitively, it seems unlikely that the level where an excellent definition of beauty resides would be situated immediately above the level of features that speak as little to beauty as rectangle shape. Does the relevance of a level of abstraction to beauty gradually increase through a series of levels, beginning with levels comprised of mundane properties, passing through levels that have a bit more aesthetic relevance, and eventually reaching a level where beauty is easily defined?

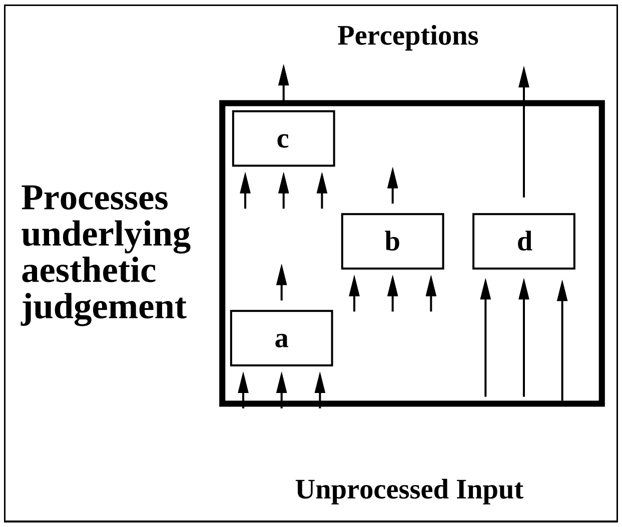

Figure 3.2: The aesthetic conceptual hierarchy, and four concepts within it. Millions of other concepts are not shown.

Figure 3.2 introduces a hypothetical hierarchy that constructs a full array of aesthetically meaningful concepts at the top, by composition of simpler concepts, built upon a base of easily-computable primitives found at the bottom. The bottom of the hierarchy is the level of direct, unprocessed sensory input. The top of the hierarchy represents the level of percepts that produce some conscious aesthetic reaction. One's overall aesthetic reaction to an input is a combination of all active outputs. A true definition of quality in art would require the construction of such a hierarchy as it exists in a human viewer of art; this would close the gap between the approaches from above and below.

The term “concept” will be used to refer to all the entities in the hierarchy, though “feature”, as the term is used by many, also applies. These entities may also be though of as “functions”, as they do entail a computation, as well as a result; “function” also calls to mind the fact that these concepts have specific inputs and outputs. All three terms apply fairly well, but for the sake of consistency and clarity, “concept” will be used throughout this chapter. Four unspecified concepts are shown to illustrate one dimension of variety. Concept (a) is a low-level concept, operating directly upon the input, and returning some result that does not amount to an aesthetic reaction of any kind, but that may be used in the derivation of other concepts. Concept (b) takes concepts like (a) as its inputs. Connecting to neither the sensory level nor the aesthetic reaction level, (b) serves a supporting role as an intermediary between low-level and high-level concepts. Concept (c) is an example of a high-level concept that relies upon concepts like (a) and (b) as its inputs. Finally, concept (d) represents another possibility — one that spans the entire hierarchy by itself, producing an aesthetic response based directly upon the input.

There are many other properties that would go into a description of any concept in the hierarchy, including the details of what inputs each concept requires. The overall connectivity may be quite complex and even recursive. It is not necessary that information should flow only in a feed-forward manner from input to conscious perception. The nature of the computation going into each concept is also important. To build upon the framework begun in Chapter 2, later discussion will consider the hypothesis that each concept in the aesthetic conceptual hierarchy is computed by either a distributed or algorithmic process, and explore the ramifications of that.

The full details of such a hierarchy will ultimately lie far beyond the scope of this work, but a few observations can be made regarding its nature. How deep is the hierarchy? How wide? What is the role of non-aesthetic perception in all of this? Why have the approaches from above and below failed? Is it because such a hierarchy is too complicated to understand? Limited information from certain subdomains of aesthetics will offer some insight regarding these questions.

3.4.1 Data on the hierarchy: Faces and bodies

Thumbnail sketches

One domain where empirical aesthetics has made some progress is the study of physical attractiveness, with separate emphases on the face and the body. This is compounded with factors of gender and sexual orientation, though attractiveness has its non-sexual side as well. In contrast to the more general case of what images are found to be attractive, this area has achieved some easy successes. There seem to be preferred ratios in dimensions of the face and body. While there are no exact parameter settings that define objectively "ideal" faces or bodies, there are nevertheless approximations and guidelines that seem undeniable. Singh [1993], for instance, finds that for a woman's body, a ratio between waist and hip circumference of 3:4 is perceived as attractive in almost any culture. It is well-established [Langlois and Roggman 1990; Alley and Cunningham 1991] that images of appealing faces can be created by averaging images of real people. Schmidhuber [1998] gives a description of a female face that is rated by subjects to be even more attractive than the faces generated by Alley and Cunningham.

What is the relationship between these findings and the putative aesthetic conceptual hierarchy? Physical beauty is a small part of the issue of overall quality in visual art. The findings of the aforementioned authors do not describe attractiveness rigorously, as characteristics like skin texture and the way that a person moves and executes facial expressions are surely a part of attractiveness. However, these findings do lie somewhere near the top of the hierarchy — they are ways in which aesthetically pleasing images can be composed from simple geometrical primitives. It would seem that these researchers have found sub-hierarchies that extend from the bottom of the hierarchy to quite high in it.

What is the nature of the attractiveness sub-hierarchies? The findings of Singh and of Alley and Cunningham are rather simple formulae applied to relatively complex input, and are thus incomplete as definitions of beauty. A simple ratio like that found by Singh does not directly indicate physical attractiveness; it has meaning only when many other properties of the image indicate that the ratio applies to the appropriate areas of a female human body. This concept is somewhere in the middle of the aesthetic conceptual hierarchy. It is quite easy to compute, but does not directly cause an aesthetic response.

Beauty and the beast

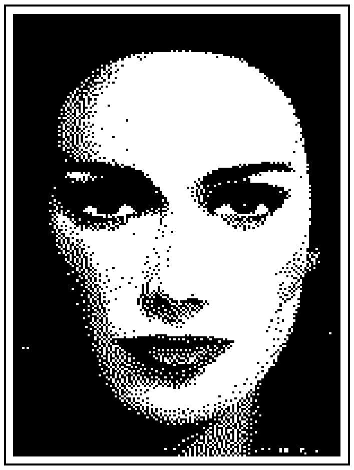

Figure 3.3: Subjects rate this face as highly beautiful.

Schmidhuber's specification of a face is more detailed than the results of the studies cited in the previous subsection — that is, it does not provide just one parameter

1	left–right symmetrical
2	distance between the eyes equals one eye-width
3	...which also equals one nose-width
4	tip of the nose halfway between chin and eyebrows
5	left eyebrow's lower edge, extended, meets right eyelid
6	left eyebrow's upper edge, extended, meets right eye's shadow's upper edge
7	line between the upper edge of the left eye and the lower edge of the right eye passes through the point midway between the pupils
8	squares of equal size are formed by lines defining upper edges of left eyebrow and left nostril, left side of nasal ridge, and left forehead
9	squares of equal size are formed by lines defining upper edge of left eyebrow, lower edge of right nostril, left part of nasal ridge, and right facial boundary
10	squares of equal size are formed by lines defining upper edges of right upper lip and the left eye's nose-side boundary, left nasal ridge, and right facial boundary
11	large diagonal squares for left and right parts of lower lip and chin
12+	large diagonal squares for "certain contours of eyes and eyebrows"
+	"many additional simple facial proportions" illustrated in a figure in the original paper
+	symmetry and many of the rules listed above entail extra rules for the other side of the face

Table 3.2: Some rules defining the shape of one particular beautiful face.

(waist-to-hip ratio) or one simple transform (grayscale averaging of multiple images) that pertain to beauty. Rather, it is a specification of several parameters — ratios between distances — that describe the shape of a face that subjects rated as highly beautiful. The image of this face can be produced by means of these parameters, in the form of something like two dozen rules (some borrowed from Renaissance artists) for the proportions of the face; information on texture and color (not provided in the paper) is required to make the eyes, lips, and other facial features appear natural. This face's description spans the aesthetic conceptual hierarchy — it is grounded in easily-computed measurements that are low in the hierarchy, and it leads directly to positive aesthetic rating, and so it has an output at the top of the hierarchy. This is therefore one example of an aesthetic concept that, like concept (d) in Figure 3.2, takes sensory-level input and produces aesthetic output. The important conclusion to be drawn from this example is that although the specification of this face is moderately complex, it does not require very long for a viewer to determine whether or not they find a particular face to be attractive. Conscious consideration of these rules is something that a person *could* attempt to undertake, but that is clearly not how faces are normally perceived. Using algorithmic mechanisms to *consciously* test a face for its beauty by checking each of these rules one by one would be a torturously slow process for a human being. The human perception of facial beauty therefore must take place by means of distributed mechanisms in a process that is parallel and fast and that produces an aesthetic reaction, all without leaving the perceiver an awareness of how it was done. One may assume that many aesthetic concepts will require descriptions that are even more intricate. For now, this serves as one example of an aesthetic concept, placing a lower bound on how complex they may be. What constitutes beauty for the aesthetic eye (Figure 3.3) is quite a beast for the analytical eye to behold (Table 3.2).

In what sense does this specification of a face amount to a definition or account of facial beauty? It is, of course, subjective, limited in generality to the subjects who participated in the study. Beyond the consideration of subjectivity, one should note that the specification addresses *what* one beautiful face is, but does not explain *why* it is considered beautiful. The paper from which the specification was taken does offer some speculation in this regard. The papers mentioned earlier that give preferred ratios of body dimensions also propose simple possible explanations for their findings — in terms of biological advantages that the preferred physical traits have. Schmidhuber offers a hypothesis about why the face he describes is preferred. For the purposes of this thesis, however, the reasons, whether innate or learned, why someone might have acquired a particular aesthetic concept are irrelevant. What is relevant is the nature of the structure of the aesthetic concept — what kind of detail is necessary to describe the concept. What Schmidhuber's work shows is that any account of "What is beauty?", even in a relatively constrained context, is bewildering in its complexity, when considered by the conscious mind. The lessons gathered from this example are important for elaborating on the framework of aesthetic perception as given so far.

This case also has suggestive relationships to other situations in which viewing an account of a mental phenomenon seems not to enrich understanding of the phenomenon. This is true in humor; jokes tend to lose some or all of their appeal when they are explained rather than "gotten" (or when the punchline is given away prematurely, or when the entire joke has been heard before). It is also true in a couple of famous paradoxes in cognitive science — Searle's Chinese Room and the hypothetical colorblind scientist who studies color. This chapter will offer a little commentary on these paradoxes in a later section.

3.4.2 Data on the hierarchy: Art criticism

Sheppard suggests (as noted above) that consideration of judgments of beauty is the key to finding a suitable definition; this is essentially the aesthetics-from-above manifesto. The point that such efforts do not lead to full elucidation of the topic has already been well established; the intent here is to use a well-chosen survey of criticism as data useful to making a rough sketch of high-level concepts in the aesthetic conceptual hierarchy. In this case, the survey of criticism provided by Barrett provides a useful "random sample" of rationales that art critics provide to support their judgments. This survey [Barrett, 1994, pp. 80–88] shows the variety in art criticism by examining the comments of many critics with regard to the same artist (Mexican painter Frida Kahlo). When opinions differ, one can try to trace the different criteria in use, because the stimulus is relatively fixed. Kahlo was chosen because the critics' opinions of her work, and the criteria offered with those opinions, differ broadly.

1	Quality of technique
2	Conciseness/simplicity of the work
3	Variety within artist's body of work
4	Not copying others; challenging tradition
5	Like/unlike some other specific artist
6	Work creates personal connection with viewer
7	Autobiographical element to work
8	Political statement made via the work
9	Status given the artist by society
10	Intensity of emotion induced in the viewer

Table 3.3: Criteria upon which judgments of art are based.

Table 3.3 summarizes the general forms of criteria that appear in Barrett's survey. The ten criteria (or types of criteria) from the survey can be considered concepts (or types of concepts) in the aesthetic conceptual hierarchy. This section strives to identify, generally, what kind of concepts in the hierarchy these criteria describe. Do they involve many inputs (high fan-in) or few (low fan-in)? Are they located high or low in the hierarchy? Do they directly cause an aesthetic response or not?

Some criteria (1–4) are concerned with structural characteristics of a work. Criterion 1 is obviously desirable, though it does not on its own make a work of art great. What constitutes good technique has been studied extensively, and is routinely taught [Lauer 1979]. The others are relational attributes of the artist's work — and again none of them inherently imply high quality. Criteria 3 and 4 may seem at first to be rather abstract and subjective. However, it is probably accurate to say that they can be determined objectively given certain levels lower in the hierarchy, and that those lower levels are where the potential for subjectivity lies. For example, one critic might say that an artist's work always invokes sadness; another critic might say that the same body of work invokes sometimes sadness and sometimes anger.

To this extent, the evaluations are subjective. No one would disagree, though, that "sometimes sadness and sometimes anger" entails variety and that "always sadness" does not. In Barrett's survey, none of the points on which critics disagree concern these criteria, although it seems possible in principle for different viewers to disagree on these matters.

The majority of the remaining criteria (5–9) involve a relationship between the given work of art and some other artwork, emotion, or event. These criteria can be used to support either positive or negative evaluations, depending upon the critic's opinion of the object to which the relationship is drawn. One must say that computing any of these is as complex as modeling a person's experience and knowledge of the world. This is where subjectivity in the evaluation of art arises, in two different ways: first, in that different viewers will relate the art to different things; and second, in that different viewers, even if they agree upon what the work of art relates to, may feel differently about that thing. Barrett produces an excellent example of the second point in that one critic implies strongly (in the backhanded unreasoned way that the anti-aesthetic commentators above wrote) that the popular celebration of Kahlo (she was quite popular in art circles at the time) was a reason to consider her overrated. Another critic revels in the positive attention Kahlo has received, and the reader is meant to approve and agree. A thorough model of this sort of criterion must combine a human-level base of knowledge and opinions of the world with a superb model of how one thing might remind someone of another. This at least matches the goals of AI to this point (though the importance of emotion here transcends the goals of most cognitive scientists) and far exceeds AI's current level of accomplishment.

Criterion 10 is the only one that does not fall into one of the above two groups, probably due to a special significance. That is, it directly entails an emotional response to the stimulus. This places it relatively high on the aesthetic conceptual hierarchy. What emotion a work of art induces may, however, also involve very basic properties, such as color, or very complex ones, such as what events are depicted and what consequences the event might lead to.

Should all kinds of criteria count?

Historically, some have argued that only certain types of characteristics of art should be used as a basis for opinions. And some of these beliefs have been extended to proposed restrictions on what artists should try to create. Tolstoy, for instance, declared that only works with morally uplifting messages qualified as art [Sheppard 1987]. It is frequently observed, often of contemporary television and films, that some kinds of pleasurable qualities do not seem to be of a lofty enough nature to merit praise, even though they attract viewers (sex, violence, and showy special effects are often denounced as baser pleasures). In contrast, a work can have high quality with respect to the intellectual product but be so disturbing (for example, with images of despair, violence, and an individual's utter loss of control) that the viewing experience is intolerable. Roger Ebert, reviewing David Cronenberg's film *Naked Lunch* (based on the William S. Burroughs novel of the same name), noticed "... the paradox of this film: While I admired it in an abstract way, I felt repelled by the material on a visceral level." [Ebert 2000].

In essence, a work can possess or lack appeal in any of a number of ways. Different individuals can vary widely in how much importance they attach to different kinds of characteristics. Some stances on what is and what is not to be counted as real art have become well-known movements in philosophy of art. For example, realism is the view that artists can only try to capture the real world; expressionism holds that the communication of emotion is the sole purpose of all art; formalism is the belief that all the criteria upon which art should be judged can be found in the form, or structure, of works of art (in literature, especially, a similar idea is called structuralism); and instrumentalism judges a work of art by the social good that it does. These are, in a way, individual taste elevated to the level of a school of thought. This is one perspective on subjectivity; another will appear later in this chapter.

Each of these doctrines has prevailed in various times and various places, but there has been a general turn towards the idea that all of them are valid, and that good art need only do *something* well, and that no single goal exists for it [Barrett 1994; Sheppard 1987]. All the criteria in Barrett's survey have their place in the hierarchy, and will be useful to constructing the framework that follows.

3.5 Aesthetic perception and the dual mechanisms

3.5.1 Perception, feature by feature

One starting point for considering judgments of style is that the perception of an object begins with the detection of elementary features — these make up the concepts at the bottom of the aesthetic conceptual hierarchy. These are primarily values along basic spatial dimensions such as height, width, and curvature — the sort of features that can easily be computed. More complex features can be created by means of composing members of the basic set. The perceptual system's ability to recognize composite features — whether acquired over the life of the species by means of evolution, over an individual's lifetime by means of long-term memory, or over the course of a task by means of short-term memory — must be applied selectively, because the number of potential composite features is boundless.[4] Many candidate

[4]The use of logical operations such as conjunction and disjunction constitutes one way to generate an enormous number of more complex features from a basic set of primitives. This is not likely to offer a complete account of the relevant human mental phenomena, but even if logical operations upon the primitives offer only a subset of all possible features, they are sufficient to show that the number of possible features is unmanageably large. A demonstration of this is the fact that with images of 1000×1000 pixels as input, there exist boolean functions such that computing them would require $10^{300,000}$ logical operations! This vastly exceeds the means of any brain or any digital computer to realize within the lifetime of the universe.

accounts have been offered of how cognitive systems may effect a composition of features [Leeuwenburg 1971; Holland et al. 1986].

The preceding comments lay out a framework in which features are detected recursively based upon a stimulus, but does not address the chronological aspects of perception — the sequence in which features are detected during a perceptual act. In the case of letters, and probably other stimuli, distributed processes as well as algorithmic processes may utilize features low in the aesthetic conceptual hierarchy, such as have been detected in the stimulus, to deduce the presence of higher-level features that are entailed by those lower-level features. The model suggested here is that both kinds of processes work continuously to generate (potentially) numerous features based on the stimulus. A model of perception (aesthetic and otherwise) that is consistent with all the evidence submitted so far suggests certain attributes, shown in Table 3.4, for the two types of mechanism in this arena.

	Distributed	*Algorithmic*
Speed	Fast	Slow
Fan-in	Potentially large	Small
Consciousness	Unconscious	Conscious
Control	Automatic	Voluntary
Affect	May create emotional/aesthetic response	No emotional/aesthetic response
Output	Pre-learned concept	Mental structure (e.g., a proposition) that binds concepts to one another

Table 3.4: Distributed and algorithmic mechanisms in aesthetic perception.

Some of the items in the table require commentary. The purpose of the "Fan-in" row is to put the parallel and serial natures of the two mechanisms into terminology befitting the framework of composition of functions. The conclusion that algorithmic processes are conscious and voluntary may seem unnatural in light of the fact that perception of many visual displays (especially simple ones like single letters of the alphabet) can usually take place without any deliberate action whatsoever. However, viewers often make deliberate decisions between perceptual strategies; this is not the norm, but it does occur. In the experiments on gridletter recognition that were reported in the previous chapter, there were instances in which a subject required nearly a minute to categorize a gridletter. It seems safe to say that these acts involved some deliberate thought. It may also clarify matters to state that "perception" is being used here in a very broad sense. For example, a viewer who is presented with a painting may look at the painting for quite a long time and make a number of deliberate choices about what perspectives (historical, religious, philosophical, etc.) to take into account in assessing the painting. It is this aspect of viewing that is deemed voluntary.

The content of the "Affect" row deserves special attention. This proposes that the kinds of concepts that lead to emotional or aesthetic responses are computed only by distributed mechanisms. The reaction to Schmidhuber's face, for instance, comes too quickly (certainly within a couple of seconds) for the amount of information that needs to be assimilated (a dozen or more rules) to take place by algorithmic means. The reaction to the face is noted to be unconscious because the viewer is not aware of any *reasons* underlying the corresponding emotional responses. In contrast, the concepts discussed by Singh and by Alley and Cunningham are easily described, but are not sufficient on their own to entail beauty (or the absence thereof). That is not to say that an easily-derived property of an image cannot produce an emotional response. Color is easily computed, and is capable of producing an emotional response [Lauer 1979]. A simple explanation is that the perception of color *is* a distributed process, but one that simply does not require large fan-in. The advantages that distributed processes have over algorithmic ones have been emphasized, but there is no reason to conclude that a distributed process *cannot* operate slowly, or that it *must* have a large fan-in, or that it *must* produce an affective response. Rather, the claims made here are merely that distributed mechanisms (and distributed mechanisms alone) *can* operate very quickly; only they *may* involve a very large fan-in; and only they *may* produce an affective response.

Perception is not purely aesthetic, of course, and the next subsection will put a theory of aesthetic perception into a framework that also explains, in general terms, perception of content.

3.5.2 Representational perception and aesthetic perception

...underneath, the unintelligible truth.

— Milan Kundera, *The Unbearable Lightness of Being*

Prior to the twentieth century, the central fact regarding visual art and literature was that they portrayed subject matter; this is still true of many forms of art, although music, typeface style, and, increasingly, visual art and poetry may lack an identifiable subject matter [Strickland and Boswell 1992; Vitz and Glimcher 1984]. Representational art does inspire feelings in the viewer, but also depicts some sort of scene, and the perception of that scene fills the awareness of the viewer.

The perception of a scene, in the feature-based paradigm, can be understood from the bottom up (though the actual perceptual process should involve some top-down influence of the sort described in, for example, the interactive-activation model of McClelland and Rumelhart [1981]). Table 3.5, which can be found at the end of this chapter, describes classes of features, in increasing order of complexity, that can lead to perception of what an image represents. Each level's features may be formed by composition of lower-level features. Overall, the levels climb out of geometrical simplicity toward the representation of complex situations and themes. The table gives each class of feature a name and gives examples of each class, as well as the means by which the feature may be computed (if possible) by existing parallel and serial computer systems. This is intended to serve as a list of the general classes of features needed to extract mere “content” from an image.

Feature Class	*Definition*	*Examples*	*Parallel computation*	*Serial computation*	*Output*
Primitives	elementary property	brightness, reddishness	perceptron	scan input	geometric form
Simplex	feature	point, line detected in field	perceptron	scan input	geometric form
Composite	composition of simplexes	triangle, circle	modular network	scan for each individual component	geometric form
Complex	computed by composition of several simplexes, composites	letters, Chinese characters, cartoon line drawings	reinforcement learning	recursion; method of moments; Examiner; others	geometric form or real-world referents
Object	real-world objects	cat, snowman	reinforcement learning?	beyond state-of-art	real-world referent
Event	real-world events	dance, war	reinforcement learning?	beyond state-of-art	real-world referent
Algorithmic Reasoning	proposition following from contents of memory	“The man loves the woman.” “Rome won a battle.”	beyond state-of-art	symbolic logic?	mental structure based on images or ideas
Distributed Reasoning	activation of concepts via associations	love, Rome	localist connectionist networks	beyond state-of-art?	learned concept
Affective Reaction	emotional reaction	joy, sadness, anger	beyond state-of-art	beyond state-of-art	emotion

Table 3.5: A hierarchy of features involved in visual perception.

According to the framework presented here, content is extracted from an image by distributed and algorithmic mechanisms as a perceptual structure is built up over the course of time. Any concept or mental structure that is built up can be assigned a place in the aesthetic conceptual hierarchy, though some concepts relate more to affect and others more to content. Throughout the hierarchy, both distributed and algorithmic mechanisms create structures in working memory for those features that have been detected. All claims made in Chapter 2 regarding the two types of mechanisms apply: algorithmic mechanisms act more slowly, can act only where fan-in is relatively small, and leave a conscious record; distributed mechanisms act quickly, can perform computations involving very large fan-in, and do not leave a conscious record of the process taking place, but can result in some sort of affective response.

The model that is proposed here suggests that aesthetic perception is interleaved with the perception of content. None of the levels in Table 3.5 is beyond the ability of distributed mechanisms to perceive, and aesthetic reactions might occur at any level of the hierarchy. The aesthetic reaction to a painting might be based upon primitive features located low in the hierarchy (such as the thickness of lines [Lauer 1979]), or due to complex, abstract features [Vitz and Glimcher 1984] such as would be found high in the hierarchy.

Each of the two mechanisms has its strengths, and any given perceptual experience will reflect this. Acts of aesthetic perception require the work of distributed mechanisms, and consequently, they do not result in a conscious record of their underlying activity. The failure, thus far, to find a good account of judgments of aesthetic quality should be attributed to the complementary weaknesses of the two types of mechanisms. It is beneficial to have both types of mechanism within the visual and reasoning systems, but that is not the same as having one type of mechanism with the strengths of both and the weaknesses of neither. Aesthetics is not easy because those mechanisms that provide self-awareness (algorithmic ones) and those that lead to aesthetic reactions (distributed ones) are mutually exclusive. This explains why the underpinnings of beauty and quality are not immediately obvious to every seeing person.

Incidentally, many, including Kant, Schopenhauer, and Edward Bullough, observe that aesthetic perceptions are "disinterested" [Sheppard 1987]. Bullough argues this point by using the way that one may experience fog at sea. He claims that there is a *practical* interest in the fog since it might cause a shipwreck, and that any aesthetic appreciation of the fog must be separate from this and is unlikely to occur at the same time, though either kind of appreciation of the fog could occur in response to a given situation. In terms of the framework offered here, one may attend to any of a large number of choices among the conceptual structures built up from the percept. For the fog at sea, an aesthetic response is one choice, a practical concern such as shipwrecks is another, and attending to the latter will preclude experiencing the former simultaneously.

3.5.3 Serial accounts of vision

This chapter has so far dealt mainly with the powers and limitations of human visual perception. Vision, including such high-level perception as is involved in aesthetic judgment, utilizes a blend of (at least) distributed and algorithmic mechanisms. At least some of the distributed mechanisms involve massively parallel processing of a wide visual field. However, when one consciously formulates, writes, or reads an analysis of visual processing, one relies upon algorithmic mechanisms, which are inherently serial in nature. The representations that algorithmic mechanisms utilize lack the sheer quantity of information needed to adequately describe the work of distributed mechanisms. Efforts to create descriptive accounts of vision must employ strategies to cope with the limited "bandwidth" offered by algorithmic mechanisms. Because solutions to this problem are less than perfect, efforts to describe vision tend to be speculative and only partially accurate. Computer models motivated by such accounts naturally suffer from the same shortcomings.

In principle, serial processing offers exactly the same computational power that parallel processing does. However, what is possible in principle is not always easy in practice, and no computer models of vision currently exist that are nearly as robust as human vision. In terms of the representational categories in Table 3.5, the best models to date enjoy partial success at the object level, but cannot handle any of the representational categories beyond that. The fact that the best attempts to model representational vision fail for more complex levels of complexity makes it more comprehensible, for two reasons, why modeling aesthetic perception has also failed. First, modeling perception of complex categories is simply hard! Handling such categories would obviously require a very complex system, and there is no guaranteed upper bound to the amount of effort required for the solution of a particular problem. Second, aesthetic reactions can depend upon successful comprehension of an image in terms of representation. If determining what an image represents is an unsolved problem, then determining appropriate aesthetic reactions to an image is also an unsolved problem.

The second reason applies only to hypothetical efforts to build models of aesthetic perception that start from the ground up, taking an image as input, managing recognition of objects and events depicted, and eventually generating the appropriate aesthetic reactions. Art critics and those who attempt aesthetics from above may seem to bypass the hard parts of the problem. Humans can take for granted their own ability to identify (often, anyway!) what objects or events a painting is intended to represent. But, while the adept human visual system allows human viewers to hurdle the hard middle levels of the aesthetic perceptual hierarchy, they still must negotiate the hard levels at the top of the hierarchy, where aesthetic reactions are really created. For those seeking a definition of beauty, the quarry is the set of processes that lead to aesthetic reactions. Each of those is potentially complicated, with high fan-in, based on lower-level features in ways of which the viewer is not even aware.

What sort of quarry is one of those processes? As the case of Schmidhuber's face shows, its details can be as alien as the end result is familiar. Given that Schmidhuber and his colleagues showed subjects hundreds of different faces before arriving at the one that the subjects liked best, there is no guarantee that finding truths (or even approximations thereof) regarding aesthetic perception is easy.

3.6 Aesthetics: Hopeless or just hard?

3.6.1 The problem reexamined

The chapter thus far has aimed at an explanation of why aesthetic perception has been such a difficult problem. Human cognition provides mechanisms capable of very high fan-in but no conscious awareness of their operation, and conversely, mechanisms that offer conscious awareness of their operation but modest fan-in. For studying aesthetics, the ideal situation would be to possess an awareness of the reasons behind one's own aesthetic reactions, but this does not exist because of the complementary nature of the two kinds of perceptual mechanism. The situation is like that of a lobster trying to crush a small object on the other side of a hole, with the larger claw being too big to pass through the hole, but the smaller claw being too weak to crush the object. Each claw has one of the two attributes crucial to the task, but neither claw has both of the needed attributes. A similar limitation exists for those who study aesthetics and, as a result, the amount of progress that has been made in the field has been limited.

Many have criticized analytical approaches to aesthetics because analytical accounts of art seem to be devoid of the vividness that is at the heart of aesthetic experience. This seems to miss the point. The aim behind an analytical approach to aesthetics is to *explain* aesthetic perception; it is not to make the analysis itself *be* an aesthetic experience. It is curious that the distinction is lost on so many commentators — analytical commentary on the subject of scuba diving does not require that the scholar be underwater while delivering the analysis or that the reader be underwater while considering it. In the cases of aesthetic appreciation and scuba diving alike, the experience and the analysis thereof do not have to take place under the same conditions. The failure to understand this seems to drive the angry reactions to analytical aesthetics by Pirsig, Schulman, and Arnheim.

So, the absence of a single "all-purpose" perceptual mechanism does not inherently forbid an analytical approach to aesthetics. It does make such an approach difficult, however. Approaches from above fail because people, however skilled they may be in their ability to appreciate art, do not have conscious awareness of the distributed processes that are vital to producing an aesthetic reaction. They are aware merely of the tracks left behind by algorithmic processes that may have served in support of that reaction. On the other hand, approaches from below fail because of the sheer complexity of the task. Decomposing the phenomena of interest (definitions of beauty, etc.) into low-level phenomena such as rectangle shape simply requires more effort than any approach to aesthetics from below has yet committed.

Although a person has no awareness of exactly what in a given stimulus triggered a specific aesthetic reaction, the person is aware that an aesthetic reaction took place (and is usually aware of what kind of aesthetic reaction — whether it was pleasure, curiosity, revulsion, etc. — it was). Although one may desire explanations of aesthetic perception that decompose definitions of beauty and style into the detailed terms commonly used in aesthetics from below, the information that comes by from-above means is essential for *finding* those explanations. Can that information (i.e., awareness of what one's aesthetic reactions are, although not awareness of the reasons for them) be used as a means to guide a program of research aimed at finding a detailed account of beauty? Such information can at least be used as a basis of communication between people. David Lynch, a renowned film director, explains how he works as follows:

> "... I don't know quite how to tell you how I think about it. But you know it *enough* to tell somebody what to do in this scene... you might use an analogy, or something that has nothing to do with it, but they get

it enough. And then when they do it correctly, they don't even know the depth of the rightness of it, but it's right." [Lynch 1997]

When one is formulating an analytical account of vision or contemplating another's account, one employs intellectual mechanisms that require attention, which are therefore algorithmic mechanisms. As a result, the rich subjective experience that accompanies the aesthetic experience is absent from the contemplation of such accounts. This has led to the fallacy that such accounts must be in error. This may be essentially the phenomenon driving other seeming paradoxes in cognitive science. One example is the paradox of the hypothetical color scientist [Dennett 1991] who has never perceived color but who understands, in complete detail, how color vision works. The riddle is: the first time that this scientist perceives color, will the sensation come as a surprise or will it be anticipated? Searle's Chinese Room scenario [Searle 1980] also discusses an analytical account of a rich cognitive phenomenon: namely, language. His argument hinges upon the intuition that a person can interact with a (hypothetical!) complete account of how to implement linguistic behavior without perceiving the richness of the phenomenon that the system purportedly models. In both cases, the root of the seeming paradox seems to be very like that of "sterile" explanations of aesthetic perception. Neither of these paradoxes can be fully dealt with in a single paragraph, but it seems promising that the distributed–algorithmic dichotomy can help resolve them by appeal to the nature of the underlying processes.

An additional dividend of the framework set up in this chapter is that it explains why it is that art critics depend so heavily upon comparing the art and artists they discuss with other art and artists already known to the reader. While a writer cannot instill an aesthetic reaction in the reader by means of analysis, the writer can do so by getting the reader to call to mind a known situation that can be reliably assumed to evoke the desired aesthetic reaction. The reader will then relate, aesthetically, to that situation in the desired fashion. Art critics, then, aim to persuade or inform by an almost Pavlovian approach, by providing associations for the reader that are not genuine reactions to the aesthetic object being discussed, but are reactions that the writer would like the reader to believe are a good characterization of the reactions that the reader would have while experiencing the object in question. A similar phenomenon is at work in the vitriolic attacks on "sterile" aesthetics such as those given by Pirsig, Schulman, and Arnheim. These attacks denounce those discussions of art that lack vividness. For emphasis, and also to avoid lacking vividness themselves, the authors of anti-sterility pieces make sure that their comments are vehement and imagistic.

3.6.2 The struggle ahead

I have not yet begun to fight.

— Captain John Paul Jones

The perception of art in all its richness and all its many forms has only been touched upon here. A complete account, if such a thing is possible, will doubtless require an enormous amount of future work. The purpose here is to use some of the insights of cognitive science (including the result of Chapter 2) to provide a general sketch of the aesthetic conceptual hierarchy. This is valuable for its own sake, but here in particular, it serves to justify the claim that Letter Spirit really does address aesthetic perception and creation.

The general problem seems very hard, to be sure, but not impossible. The framework put forth in this chapter suggests that some of the roadblocks are merely hurdles that might be leapt with greater effort. Even though aesthetics from above and aesthetics from below have not met in the middle of the aesthetic conceptual hierarchy, a couple of paths that go nearly all the way through have been found. Even if introspection can never tell a viewer the reasons behind aesthetic judgment, there is hope that the position from above may provide the high ground for reconnaissance that guides a successful effort from below.

Machotka's four reasons why aesthetics from below has not succeeded seem a bit less potent in light of this chapter's discussion. Indeed, although it is hard to experimentally determine general aesthetic reactions to stimuli, it is not impossible. Although it is hard to know where to begin, it seems possible that a successful approach is out there to be found. Schmidhuber's research on facial attractiveness gives an example of how the underpinnings of some very complex aesthetic judgments can be uncovered; this is a tiny piece of the problem of quality in all of visual art, but it does suggest that with many followups, the number of topics in aesthetic perception that are understood might start to accumulate. The first two hurdles described by Machotka are empirically based; the story so far suggests that they are indeed difficulties, but it is not obvious that they are insurmountable.

The remaining objections of Machotka seem even less worrisome. The fact that the importance of technical instruction in art has waned over the centuries pertains only to a certain level of abstraction (or lack thereof). Vitz and Glimcher [1984] argue that the level upon which art has focused has moved, over the centuries, from more concrete properties to more abstract ones. Perhaps on some level of abstraction — maybe on a meta-level of striving to be abstract, or to be original by shunning tradition — artists are as imitative as ever. The fourth difficulty that Machotka poses — that any explicit characterization of quality is something that artists will strive to avoid — presupposes a certain simplicity to the characterization. Suppose that an understanding of artistic quality filled several volumes and could be modeled on a computer only by a program millions of lines long; such an account would be far too complex for anyone to mentally "run", and for that reason it would be impossible for an artist to use it to figure put which types of beauty should therefore be avoided. A hypothetical computer program that rendered humanlike aesthetic reactions to art would not ruin the possibilities of art any more than a human critic does so now. The relationship between an artist and the hypothetical computer program rendering humanlike aesthetic judgments would be no different from that between an artist and an audience today. Indeed, the artist could try to avoid what was pleasing to the computer at some earlier point in time, but this simply sets into motion an evolution of standards in which the artist and the viewer (computer or otherwise) might both take part. This is very much like Boden's account of the evolution of music over the centuries, and Vitz and Glimcher's account of the evolution of visual art. Just as the output of the program would not destroy art, neither would the details of the program's structure and processing. The details of the hypothetical program's aesthetic reaction would be so lengthy and elaborate as to be meaningless when viewed in detail, just as the rules in Schmidhuber's account of a beautiful face do not seem to have aesthetic relevance until the face is instantiated as an image and presented to the eye.

Subjectivity in aesthetic judgment, in light of the framework of the aesthetic conceptual hierarchy, should be interpreted as a difference across individuals in terms of which concepts are employed during aesthetic perception. Earlier in the chapter, a partial account of subjectivity mentioned that individuals may have broad differences as to which things they consider desirable, and what they consider admissible as art. A person with a strong stomach may enjoy a thought-provoking film that contains violent or sexually explicit images. Someone else may appreciate the abstract issues but be unable to overcome a sense of disgust engendered by the images in the film. A third person might enjoy the raw images, and thus the film, but not even be able to perceive the abstract ideas being put forward. This is one example of myriad ways in which individual differences can lead to subjectivity in judging art. Future studies of aesthetic perception should devote a great deal of consideration to development and to individual differences.

The discussion in this chapter may shed some light on the seeming paradoxes behind aesthetic perception, although of course it does not eliminate the immense amount of work that would be required to produce a complete account of aesthetic perception. The failures of the past may be entirely due to the insufficient amount of effort put into the undertaking so far. While a talented person's life work, or that of a small team of people, may seem grand on one scale, one of the lessons of the twentieth century is that some projects simply require more resources than that.

3.7 Letters' spirit

An account of how letters are represented in terms of roles was presented in Chapter 1, and an account of the processing underlying that was offered in Chapter 2. The framework laid out in this chapter allows an account of how style perception occurs at the same time, using both the letter level and the role level.

The effort to define style has run up against many of the same problems as the effort to define beauty. There are accounts of style, but the definitions are partial and conflict with one another, and it would be exaggerated to claim that there has been anything more than limited success. However, many definitions that have been offered corroborate the choice of stylistic properties that Letter Spirit uses (which were introduced in Chapter 1).

Nöth [1995] reports a variety of theories on style, with those theories apparently at odds with one another. On one hand, Fricke proposes that in any medium there is a norm, and that style is a deviation from that norm [Fricke 1981]. The commonality between this and Letter Spirit's norm violations (based on violations of norms of roles) is obvious. Others, like Todorov and Zemsz, reject the idea of deviation from norms and say that style is "the internal characteristic" of a work — in other words, that it is something that is simply noticed on its own, having nothing to do with *a* *priori* expectations [Todorov 1965; Zemsz 1967]. This is true of Letter Spirit's motifs and abstract rules.

The sense of contradiction between Fricke on the one hand and Todorov and Zemsz on the other is akin to many disputes that Sheppard finds frequently in the literature on beauty and art. Although a general phenomenon such as beauty or style may arise in more than one way, a given scholar in the field frequently becomes a champion of one particular way in which that phenomenon might be realized, assuming that only one way is correct, and buttressing all arguments with supporting examples, but never showing that counterexamples supporting the other views do not exist. This is, of course, very like the conflicting accounts of letter recognition cited early in Chapter 2, each side finding support for its own model and thus discounting the possibility of the other. Working models of aesthetic perception and creation may help to undermine the surprisingly common assumption that every cognitive phenomenon is based on just one underlying mechanism.

In addition to the views of Fricke, Todorov, and Zemsz, there are accounts of style that describe it as the addition of properties to the whole, or the choice between possible alternatives [Nöth, p. 343]. Koch calls style the *semantic differential* between the overall meaning of a work and the core message [Koch, 1963] — sort of a fringe-benefits package that comes with a work of art. Riffaterre [1959; 1971] calls style an element of surprise. Briefly, there are many accounts of style. There is an encouraging relationship between many of them and one or more of the stylistic properties in Letter Spirit, but the relationship is often vague, because the accounts are, in the first place, usually fairly vague. One often senses that the author has a strong feeling for what style is but has done a poor job of finding the words to express it. This undesirable state of affairs is, in fairness to these authors, what the framework based on the distributed–algorithmic dichotomy predicts. In Letter Spirit, stylistic coherence across all letters in a gridfont is the goal. Three kinds of stylistic properties serve as the basis for judgments of coherence, and they have already been introduced. The representations and processes behind the implementation are described in detail in Chapter 6.

CHAPTER FOUR

Creativity

4.1 Introduction: What is creativity?

Creativity is a subject that has long been of interest, but whose nature has been only murkily understood. This chapter has two main purposes. The first part of the chapter considers how the term "creativity" has been used in different ways by different scholars and seeks to put the rest of the chapter on a solid footing by offering a few clearly-defined terms upon which the subsequent discussion can be based without the ambiguity inherent in a term like "creativity". The second part of the chapter describes mechanisms that underlie the different phenomena called "creativity" and illustrates how certain strategies can help computational models of these phenomena produce higher-quality output.

4.1.1 How "creativity" is defined

In terms of history, the study of creativity has been similar to that of beauty in many ways. For both studies, the interest dates back to antiquity (the deliberate use of the golden mean as a principle in architecture goes at least as far back as the fifth century B.C., being exemplified in the Parthenon), but neither of the two phenomena has yet been understood in a complete and satisfactory way. Both phenomena have been of interest to scholars in the humanities and the sciences alike. The issues differ, though, with respect to the sort of difficulties they involve. In the case of beauty, the various discussants seem to agree upon what phenomenon they are all investigating: that there is a characteristic of objects (also dependent upon subject, most grant) that produces a positive affective response. (They differ, of course, in what they think this characteristic is.) In contrast, the term "creativity" is used in many senses. Some studies (for example, [Hennessy and Hinkle 1992]) are concerned with high-level commentary on social and environmental contexts. Others sharing that high-level focus have business, management, and organizational productivity as their primary interest. Neither of these types of studies pays much attention to underlying cognitive mechanisms. There is, however, a large body of work on cognitive approaches to creativity [Boden; Roskos-Ewoldsen et al.; Sternberg]. However, even those studies of creativity with a cognitive outlook do not all agree upon exactly what it is they are focusing on. An examination of cognitive-oriented studies of creativity shows that they refer to many phenomena, not one. Definitions of "creativity" differ slightly from one author to another. When examples of "creative" behavior are offered, the characteristics that seem to be critical to the creative nature of those examples differ from one author to another, suggesting that some consider creativity to be a broader phenomenon, while others have a narrower definition in mind. Table 4.1 shows the characteristics that seem to be part of most authors' notion of "creativity".

4.1.2 Untangling the definition

A narrow definition of creativity would stipulate simply that exactly those acts that fulfill all of the characteristics in Table 4.1 are instances of creativity. Many scholars, however, use a looser definition. Boden, for instance, uses the term in a way that

Productivity	Some product is generated.
Originality	The product of the creative act is original in a fundamental way.
Value	The product of the creative act is highly valued (aesthetically or practically).
Talent	Creativity comes from people with great talent.
Opacity	Introspection does not inform the creative person very well about their own creative process.
Mystery	Creativity is by its nature impossible to understand.

Table 4.1: Characteristics frequently associated with creativity.

incorporates only some of the six characteristics as definitional[1] and considers whether or not "mystery" is an integral part of creativity as an open question to be explored [Boden 1990]. Langley and Jones distinguish a phenomenon they call "insight" — sudden and unexpected discovery that entails opacity — as a special kind of creativity. Langley and Jones' definition of "insight" is probably what many others mean by "creativity". It is certainly an apt description of the most prototypical and most famous (bordering on hackneyed) examples of creative acts — Kekulé's conception of the ring structure of benzene being particularly oft-cited. By defining insight as a subset of creativity, Langley and Jones free up the latter term to have a broader sense. Minsky [1985, p. 80] broadens "creativity" almost to the point of discarding it, and opines that there is not "much difference between normal and 'creative' thought." Hofstadter [1985, p. 527] subscribes to a similar thesis: "Creativity is part of the very fabric of all human thought, rather than some esoteric, rare, exceptional, and fluky by-product of the ability to think, which every so often surfaces in places spread far and wide." Minsky and Hofstadter, then, define creativity in a way that does not require all of the six characteristics.

The cursory survey of definitions of creativity offered above demonstrates that

[1]She makes even finer distinctions by exploring, for example, how the term "original" should be defined.

there is variety in the definitions that have been proposed. Against what standard should candidate definitions be judged? Each of them denotes some subset of human behaviors, but that is a rather empty claim; any arbitrary set of characteristics (e.g., things done on a Tuesday that involve a coconut) does that much. The approach taken in this thesis is to understand the mechanisms that underlie cognition, so the question is whether or not creativity (according to *any* of the standard definitions) corresponds to the activity of some real mechanism in the mind. This chapter will develop the argument that the standard definitions of creativity do not have such a basis. There are reasons why the set of characteristics in Table 4.1 tend to co-occur, but it is argued that there is no reason why they *must* do so.

4.1.3 Bases for a possible definition

Since this chapter seeks to develop a definition of "creativity", it will not begin by providing one. The discussion will instead begin by focusing on the candidate component traits of creativity (namely, the characteristics in Table 4.1), and will treat "creativity" as a term with an established meaning only after certain issues mentioned above have been dealt with.

Most of the characteristics listed in Table 4.1 (the exception is productivity) are not boolean. For instance, it is not the case that the product of a particular act either has value or does not; rather, the amount of value that one may ascribe to the act ranges along a continuum. As a result, one may define creativity in terms of graded characteristics in one of two ways. The first would be to have a definition that specifies a threshold for each graded characteristic, so that only acts that exceed the specified level (for each characteristic) would be considered creative. With a definition of this kind, it might be possible to denote exactly those acts that are the product of a specific cognitive mechanism, if one exists. The second possibility is to allow that the definition of "creativity" itself be non-boolean, one that establishes no sharp cutoff between acts that are creative and acts that are not. In this case, the definition would correspond only loosely to any specific cognitive mechanism or mechanisms that may support "creativity". Having a definition based upon graded characteristics forces the definition of the phenomenon as a whole to take one of these two forms. Later discussion will show that satisfactory definitions of creativity are of the second kind. Value is intimately intertwined with creativity, but is not a telltale indicator of the work of any special cognitive mechanism.

Originality, like value, is non-boolean, but is special in requiring that one make a distinction between what Boden calls personal-creativity, or *p-creativity* (acts in which the result is original as far as the actor knows), from humanity-creativity, or *h-creativity* (acts in which the result is truly original) [Boden 1990]. The latter is naturally a subset of the former. In general, one may say that works and discoveries in the latter group are more original than those in the latter, but whether or not any particular work qualifies as h-creative may depend in part upon chance. This chapter focuses upon the p- version.

The role of opacity in creativity is consistent with the finding of Chapter 3 that the underpinnings of quality can be hard to identify, but implies the further assertion that not even the *creator* of a work of art may be able to identify why the work achieves a particular quality. David Lynch, the film director mentioned in the previous chapter, is outspoken about the fact that he is unaware of what it is that he finds good about his own work. He elaborated upon this a bit while commenting on his film *Eraserhead*.

> Certain things are just beautiful to me, and I don't know why. Certain things make so much sense, and it's hard to explain. I *felt Eraserhead*, I didn't think it.

This extends the issue of opacity as a trait of certain kinds of perception to a trait of certain kinds of production. It also reinforces the idea that aesthetic activity, in contrast to other forms of cognitive activity, is related to emotion. All of these points suggest that something like the distinction between distributed and algorithmic processes seen in the preceding two chapters recurs in production.

Also worth discussing is what Langley and Jones call “insight”. For the purposes of this thesis, insight will regarded as a special kind of creativity, different from creativity in general in that its definition depends upon the process (the result comes suddenly and unexpectedly), rather than on the product. It seems possible that two people working on the same problem could find the same solution, with one solving the problem by insight, and the other by the methodical application of a known method. This contrast is at the heart of discussion found later in this chapter.

4.2 Production and the two mechanisms

4.2.1 Importing the dichotomy

In the previous chapter, the characterizations made in Chapter 2 concerning distributed and algorithmic mechanisms were used as a framework for studying aesthetic perception. It was concluded that acts of aesthetic perception, although they may involve both types of process, depend upon distributed processes as the basis for their aesthetic nature. A similar argument is offered here regarding creativity. It will be assumed that the distributed–algorithmic dichotomy, demonstrated so far only in the context of perceptual processes, carries over to productive processes as well.

This framework explains many aspects of established views of creativity. Some of the characteristics attributed to creativity have been established (or suggested) as traits of distributed perceptual processes, and this raises the possibility that creativity has its origin in distributed mechanisms. For example, distributed perceptual processes leave the perceiver without any awareness of how an answer was derived, and this is similar to the opacity that accompanies creativity. Also, many have noted the role that imagery frequently has in creativity; imagery is in turn correlated with distributed processes in Sloman's table [Sloman 1996; Shepard 1978]. Distributed processes have a greater fan-in than algorithmic processes; simply by tapping more information as input, they have a greater prospect of making a novel combination as output, and thus a greater prospect for originality.[2]

At the very least, distributed processes have certain broad attributes that one would expect to be true of the basis for creativity (as well as of Langley and Jones's "insight"). In order to explore the possible connection more closely, the next section grounds its discussion in the processes and mechanisms that are involved in acts that one may choose to call creative.

4.2.2 Distributed versus algorithmic, across the board

The position argued for in this chapter is that it is profitable to discard the notion of "creativity" and adopt a framework in which *productivity* is the general phenomenon, with the characteristics of interest for productive behavior being the nature of the underlying processes, as well as the level of quality that results from particular circumstances.[3] A later section in this chapter will show that the distributed– algorithmic framework essentially deconstructs the standard notion of "creativity", as a consequence of productive behavior being based upon those two types of underlying process. Then, a subsequent section will evaluate strategies for production in terms of their resulting quality. The current section introduces the distributed–algorithmic

[2]The integration of many elements from different sources suggests what Koestler referred to as the "bisociation of matrices" [Koestler 1975].

[3]While all creative acts, by anyone's definition of "creative", are productive, not all productive acts meet more stringent definitions of "creative". For example, adding two integers is productive, but would be considered by few to be creative.

framework in the context of productive behavior by examining the research on expertise in the domain of chess.

The most common AI approaches to games such as chess consider game-playing to be essentially a matter of search. Because game trees tend to be large, the usual strategy is to examine only a small portion of the tree and to evaluate the branches that emanate from each of the legal moves that follow from the current board position. All of the board positions reachable within a few moves are rated for their desirability by a static evaluation function, and the AI program selects its move for the current turn in such a way as to try to bring about board positions that the static evaluation function rates favorably. This sets up a classic tradeoff. With a perfect static evaluation function, a search algorithm would only need to look one move (one "ply" — one "move" usually means a move by each player) ahead. At the other extreme, if the algorithm could perform a full search of the entire tree, the static evaluation function could be trivial — merely determining, for any node in the game tree, if either player had won. For games as complex as chess, neither of those extreme strategies is feasible, either for a human or for a digital computer.

A clear contrast between human and computer chess is that humans consider far fewer board positions than do the most successful chess programs. To achieve a level of play comparable to that of a computer program, a human player must have a dramatically superior evaluation function (and/or a dramatically superior method of selecting which moves to evaluate) [Nozaki 1990].

How is human skill in chess acquired? Once the rules are learned, a novice may play by examining algorithmically board positions arising from future moves, applying some crude evaluation function to the resulting short list of board positions. However, if a player has no experience to suggest which moves are worthy of focusing on, time constraints will result in extremely shallow lookahead, and consequently poor play. However, a human beginner at chess has excellent prospects for improvement.

Surely chess experts are not born with superb skills in evaluating chess boards. An account of how humans acquire chess expertise is very instructive. Simon and Chase [1973] found that chess novices see a chess board in terms of individual pieces, while experts see the board in terms of patterns of several pieces at a time. This suggests that beginners examine boards in terms of pieces and find patterns algorithmically (with the rules of chess as primitive operations). However, for humans, an automatic consequence of *playing* chess is *learning* about chess. Chess patterns involve large numbers of pieces and squares on the board, and this is just the sort of high fan-in that is characteristic of distributed processes. Distributed processes also allow for generalization, which is important, because chess games are not given to exact repetition once one is past the first few moves.

There are two kinds of useful functions that distributed learning during the playing of chess can fulfill: first, for purposes of pruning the search tree, learning which possible moves are worth considering, and second, providing a static evaluation function. Both of these tasks can exploit advantages of distributed processes — not only high fan-in and generalization, as noted above, but also speed. To the extent that expertise in chess is a matter of rapidly evaluating the board, it appears that several factors make distributed processes better qualified for the purpose than algorithmic approaches to the same task.

This does not show that algorithmic processes play *no* role in cognition underlying chess-playing; there can be a variety of mental strategies involved. Additionally, in terms of improving through experience, there may be a variety of kinds of memory involved as well (explicit and episodic, as well as implicit).

A point from Chapter 2 to return to is the question of whether or not weighted feature bundles are adequate models for distributed mechanisms. In chess, it seems impossible that such a simple approach, combined with contemporary computer processing power, could produce the level of play of today's top computers, which is about the same as the top human players. This is consistent with the conclusion of Lakoff that weighted feature bundles are not sophisticated enough to serve as a model for human categorization.

The next section examines productivity in mathematics, and many of the observations just made will be seen to apply to that domain as well. One may expect that a mathematical genius also acquires a repertoire of techniques through experience, and that the development of a mixed distributed–algorithmic method can apply there as well.

4.3 Genius: The tortoise and the hare?

The standard view of creativity suggests that the creative person taps into an inner process that fulfills all the characteristics in Table 4.1. It is so essentially mysterious that earlier thinkers have likened it, in the form of muses, to an external, godlike force that enriches the thought process. This section will argue that this popular concept of creativity is not due to a specialized mechanism for creativity, but is a byproduct of the distributed–algorithmic framework established in the prior two chapters. Observations regarding skill in mathematics, as it ranges from novice to expert to genius, will be used to suggest that creativity results from the correlation of several behavioral traits that are not inherently coupled together.

Part of the mystique that surrounds "creativity" is the implication that what distinguishes The Creative Person from an ordinary person is not merely a matter of extent. The skill of The Creative Person is often portrayed as not merely the advantage a hare has over a tortoise, but more like the advantage a hawk has over a tortoise — a qualitative difference in approach and ability. This section frames the contrast between The Creative Person and the ordinary person not in terms of their personal attributes, but rather in terms of the different ways they explore the same domain. The conclusion will be that the contrast is not so much like a hare running faster than a tortoise over the same track as it is like one contestant taking a different (and shorter) track than the other.

4.3.1 The hare

A classic and heralded example of The Creative Person was Srinivasa Ramanujan. Lacking any formal credentials or even nominal academic success in his native India, Ramanujan parlayed his unusual ability in mathematics into a brief, noteworthy career in England. Having devoted many hours per day for much of his young life to playing with numbers, Ramanujan began a correspondence with Trinity College professor G.H. Hardy. Ramanujan's first letter exhibited a series of his discoveries as testimony to his mettle. One of the gems in his letters ([Kanigel 1991, p. 168]) was that if:

$$u = \cfrac{x}{1 + \cfrac{x^5}{1+\cfrac{x^{10}}{1+\cfrac{x^{15}}{1+\dots}}}}$$

and

$$v = \cfrac{x^{\frac{1}{5}}}{1 + \cfrac{x}{1+\cfrac{x^{2}}{1+\cfrac{x^{3}}{1+\dots}}}}$$

then

$$v^5 = u\frac{1-2u+4u^2-3u^3+u^4}{1+3u+4u^2+2u^3+u^4}$$

This set of discoveries convinced Hardy that the Indian possessed special talent. The one presented above was particularly strange and beautiful. During their correspondence, Ramanujan was unable to tell Hardy how he had derived it, saying that his means of proof were hard to explain. It was proved by an English mathematician eight years later (sadly, after Ramanujan's early death), in a ten-page proof that seemed sure to be quite different from Ramanujan's approach, as Ramanujan did not use traditional methods of proof. His results were usually offered without the detailed derivations that English mathematicians had come to expect of one another's work.

It was no easier for Ramanujan to explain his methods to English mathematicians in person. One problem, phrased as a puzzle involving house numbers along a street, Ramanujan solved very quickly. His solution, like the previous example, involved continued fractions. Ramanujan described his approach as follows:

> Immediately I heard the problem it was clear that the solution should obviously be a continued fraction; I then thought, Which continued fraction? And the answer came to my mind. [Kanigel, p. 215]

Hardy gave a summary that corroborated Ramanujan's assessment of himself:

> All his results, new or old, right or wrong, had been arrived at by a process of mingled argument, intuition, and induction, of which he was entirely unable to give any coherent account. [Kanigel, p. 216]

As this comment points out, many of Ramanujan's wondrous results were not, in fact, correct. This certainly says something about his method. However, those of his results that were correct, including the two mentioned here, attest to an ability far beyond normal. In fact, his particular strengths were beyond that of the best mathematicians of the time, as well; conversely, *their* abilities — particularly the ability to produce rigorous proofs — were beyond him.

Ramanujan's work relied in large part upon insight (in keeping with Langley and Jones's definition of the term, offered earlier). He is not the only mathematician ever to have relied upon it, but the extent to which he did was exceptional. His work, such as it is documented, shows that he did employ *some* mundane steps along the way to his masterpieces, but striking, lightning-bolt-sudden steps would often carry him from one point in his derivations to another. The results that Ramanujan leapt to in one step would often take another mathematician, even a talented one, pages of work to derive by traditional means. Even after Ramanujan and the English mathematicians had had years to get to know one another, neither had absorbed much of the other's art.

4.3.2 The tortoise

In contrast to Ramanujan's work, an example of a productive act that seems to require nothing like the prototype of creativity is a simple algebra problem — solving a quadratic equation in two variables for y.

Step	*Reason*	*Motivation*
$(y+1)(y-1) = 4[x^2+x]$	Given	
$y^2 - 1 = 4[x^2+x]$	Multiply, left side	Get to only one y term
$y^2 - 1 = 4x^2 + 4x$	Multiply, right side	Get x terms on same level
$y^2 = 4x^2 + 4x + 1$	Addition, both sides	Isolate y term
$y^2 = (2x+1)(2x+1)$	Factor, right side	Make right side like left side
$y = 2x + 1$	Square root, both sides	Get y term down to just y

This problem is as short and simple as its solution, and while the solution given above demonstrates productivity, it lacks the other qualities in Table 4.1. It is not original. It has almost certainly been posed and solved many, many times before. The individual steps and the motivations behind them are quite ordinary. It is not of high value, although it is not of particularly *low* value, either — it is simply a correct application of algebra. While either the final line or the entire mini-proof may perhaps be of some use, such minor victories are mundane, and bright ninth-graders can be expected to solve problems like this repeatedly, without fail, but also without glory. The value criterion calls for a higher significance to the feat. Obviously, the talent criterion is not met, either.

It is interesting to consider in what ways opacity and mystery apply here. Most observers with a bit of algebraic skill would say that the proof contains little that is puzzling. The reasons for each step would seem clear even if they were not given in the second and third columns. As is the case with the most basic perception of visual art, that of merely identifying what objects are being represented, this proof might seem anything but opaque or mysterious to an intelligent evaluator. In terms of a complete fine-grained cognitive and computational description, however, basic perception of objects and simple algebra proofs do elude complete understanding. A brute-force algorithm can solve the problem, but not the way a human does. The steps and reasons in the algorithm's proof might be like the human's, but the algorithm would not be driven by the same motivations. The reasons behind the reasons would be hard to produce. Is the use of basic algebra a mystery or not? This is a question to which this chapter's discussion will return.

4.3.3 The race

Ramanujan's work is especially impressive in the context of more basic and conventional mathematics. A talented but less remarkable mathematician could, in principle, arrive at the same formulas that came to Ramanujan in a flash by means of a much more circuitous route. As was noted above, a skilled mathematician validated the Ramanujan result displayed earlier, by means of ten pages of algebraic manipulations (plus the unwritten thoughts that led to many of the decisions demonstrated throughout those manipulations). A skilled mathematician tends to operate in a bit more free-wheeling manner still than that used in the short proof above. Expecting a readership of other skilled mathematicians, one can skip many intermediate steps of trivial complexity (where the scope of "trivial" depends upon level of skill). A meticulous, step-by-step proof of the "v^5" discovery would be longer still, perhaps in the dozens of pages. It is unlikely, however, that a mathematician who could operate only at that level would ever solve such a difficult problem. To make a more realistic contrast, Figure 4.1 provides an abstract representation of the work of three kinds of mathematicians. In the figure, each vertex on the lattice stands for a mental representation of an equation or a thought or image otherwise relevant to the proof, and from every vertex, basic operations specify single steps that can be made between representations, indicated schematically by the segments that connect vertices.[4] In the simplified schema shown here, each vertex allows only four possible operations leading to another, while in reality, the number would be higher — potentially much higher. Each path shown in the figure is the means by which a mathematician has taken a set of givens (represented by some vertex along the bottom edge of the figure) and derived some result (the terminal vertex in each path). The path on the left shows the work of a novice on a short, simple proof. The path in the middle shows the work of an expert, a skilled mathematician on a slightly harder problem. Finally, the path on the right shows the work of a genius producing a result of great value.

The novice proceeds methodically through the short proof, taking, one at a time, steps that have been learned from texts and teachers. The expert skips some steps, as a skilled athlete might in running up stairs. An expert mathematician might easily accomplish a proof like the one given earlier in two steps instead of five, operating on both sides of the equation at once, or compounding several steps on one side, and writing down *that* result down as a kind of super-step. These steps may puzzle the true novice, but the ability to double-up and triple-up on steps comes to just about anyone who practices algebra. In the diagram, these steps allow the expert to make

[4]Naively, one could assume that each vertex corresponds to an equation and every step to a mathematical (e.g., algebraic) derivation. It is more apt to be as general as possible and imagine that the steps in this discussion can be of any kind whatsoever, whether involving formulas, imagery, or any other cognitive representation that a mathematician may conjure up.

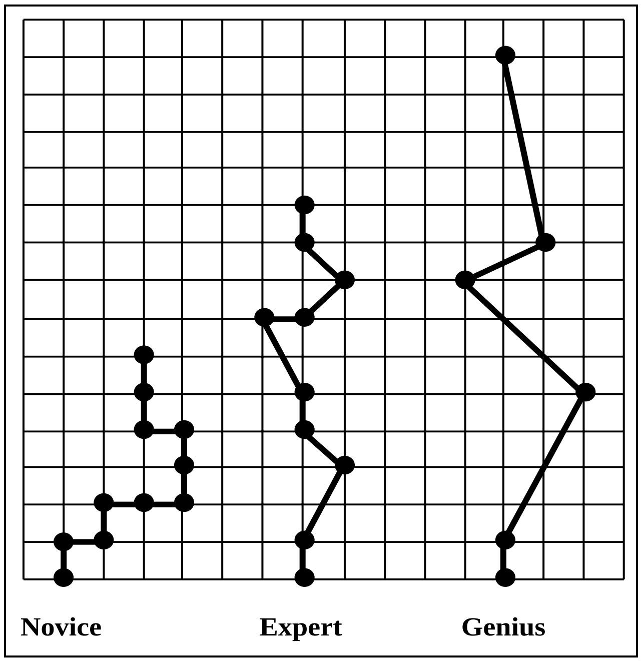

Figure 4.1: The work of three kinds of mathematicians, in schematic space.

some moves from a vertex to a non-neighboring vertex. This kind of superiority over the novice is clear, but not particularly mysterious.

The genius might be said to stand in the same relationship to the expert as the expert does to the novice. Large leaps occur, which would require an enormous number of steps by the novice, and even many steps by the expert, to achieve the same progress. From that standpoint, the difference between the genius and the expert is merely one of extent, not of a categorically different mechanism at work in the genius. However, there are some differences. An expert could, if called upon to do so, take a proof made with some of the "skipped steps" and insert what was skipped. The expert is not forced to move forward by big leaps at a time, but is able to do so, and will usually choose to do so. The case of Ramanujan shows that a genius may not be able to reply in kind. Ramanujan never fully mastered the skill of constructing rigorous proofs like his expert colleagues in England. As a consequence, he was also unable to provide the more painstaking, novice-level pathway to his results.

It would be easy at this point to end the analysis and declare Ramanujan's genius an uncanny wonder — to say that while others walked and ran about the mathematical landscape, Ramanujan flew. As Ramanujan was not able to produce an explanation of where his wondrous ideas came from, distributed mechanisms could be evoked, vaguely, as the explanation. Distributed mechanisms, insight, bisociation of matrices — all three offer just a tease of a real explanation. Is a better explanation at hand? A fine-grained analysis of how Ramanujan came up with his particular results is unfortunately not at hand, nor is a recipe for replicating such discoveries. It is possible, though, to demythologize his talents somewhat, not so much by lowering one's esteem for Ramanujan as by correctly identifying how the talents of a skilled (but not genius-level) expert compare with those of Ramanujan. The following sections argue that Ramanujan did not so much move about the landscape faster than others as employ specialized shortcuts that would help him excel in precisely those races that finished where his shortcuts happened to lead.

4.3.4 The racetrack

The abstract mathematical space in which all the mathematicians operate has been referred to — a few times now — as a landscape. This may conjure up images of fields, streams, and foliage, but it is also intended to evoke the technical use of the term, common in sub-branches of cognitive science — especially genetic algorithms — and dating back to its applications in biology by Sewall Wright in the 1920s.

The analysis thus far has been based on the image of a landscape, and has suggested that a novice, expert, and genius move about via leaps of different size. This is surely a common intuition that comes when one encounters thinkers, in any domain, of different levels of ability. As it turns out, however, this is not a very evenhanded account of things, because it defines the landscape in terms of one set of operators (the novice's) and then assesses the behavior of systems making use of other sets of operators in those terms. The result is that the latter systems (especially, here, the work of the genius) seems to skip about the landscape.

An important and perhaps surprising observation regarding this result is that it is essentially tautological. As was shown by Jones [1995], different sets of operators define different landscapes. On a landscape defined by one operator, the steps allowed by a second, unrelated, operator, are almost guaranteed to be large leaps across the landscape. This is not an argument for the superiority of the second operator. In fact, the situation is symmetrical if the steps of the first operator are measured on the landscape defined by the second. Jones explores this effect with the operators commonly used in genetic algorithms, where bitstring crossover is commonly perceived as an operator that leaps about a landscape whose definition, it is easily forgotten, is expressed in terms of single-bit mutations. As it happens, single-bit mutations leap about a landscape defined in terms of crossover in much the same way.

In an significant way, the relationship between the work of Ramanujan and the work of the expert is symmetrical. While there exists a perspective in which the work of the more conventional expert is a plodding, stepwise exploration of the abstract space, while the work of Ramanujan leaps about miraculously, there is also a way of rendering the abstract space such that the roles are reversed. Figure 4.2 shows how the work of the expert and the genius might be represented if the unit steps between vertices are the ones that the genius employs — like those in which Ramanujan immediately saw that a problem's solution should be a continued fraction, or in which the choice of which continued fraction simply came to his mind.

In the case of Ramanujan, this figure seems to be quite accurate, as the feats of a conventional expert were indeed beyond him. For a genius like Gauss, one might argue that this diagram is unfair, that the genius could operate *either* way. That is a fair observation — a large set of operators makes a great variety of accomplishments possible. The example of Ramanujan, however, shows that when the sets of operators

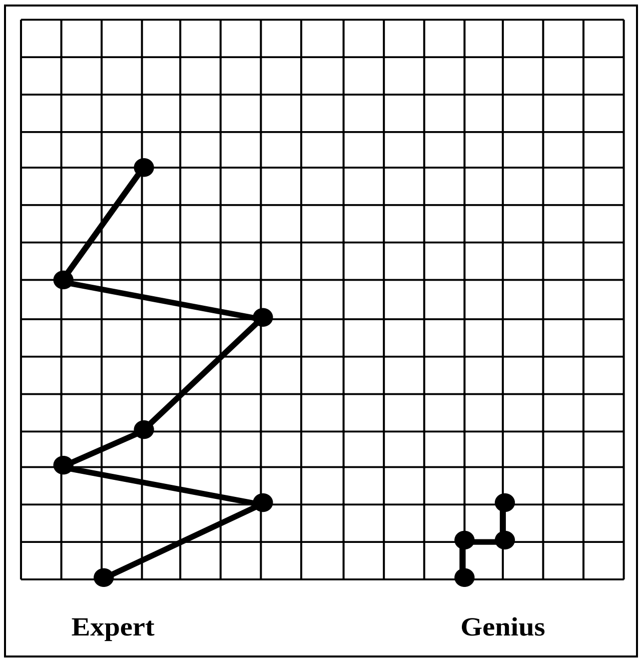

Figure 4.2: The novice and the genius, in expert-space.

available to a pair of mathematicians are relatively distinct, then each may be unable to perform what the other finds basic.

4.3.5 Finding paths and following paths

The great wonder of Ramanujan was that he was self-taught, to a considerable extent. He spent countless hours playing with mathematics, making small discoveries and acquiring skills that would let him later make big discoveries. The fact that he was self-taught and that he was able to accomplish such original work may not have been unrelated. The situation recalls the distinction Sloman made between his two systems of reasoning, back in Table 2.1. To wit, the reasoning that is Sloman's equivalent of distributed mechanisms has its roots in personal experience, while his equivalent of algorithmic mechanisms comes from language, culture, and formal systems. A number of signs, then, seem to indicate that Ramanujan acquired, by means of a great deal of personal exploration, distributed mechanisms for mathematical derivation. The techniques had a high fan-in, as they operated on, at the very least, the many elements of large formulae[5] all at once. The processes underlying his techniques occurred so seamlessly and swiftly that he was left with little or no understanding of how they took place. He could not explain them to others, and they apparently were quite original, as none of the Western mathematicians he later met had such skills. In summary, his techniques exhibited productivity, originality, opacity, value, and talent; the issue of mystery has yet to be explored.

The work of a more conventional mathematician is not altogether different. Exercising a step such as factoring a polynomial amounts to the execution of a learned pattern. The procedure can be learned and applied mechanically (when the values of the coefficients and exponents make it possible). Most of the steps in basic algebra proofs are of this nature. However, some are a bit more complex. Some steps in the proof offered above have simple descriptions, like "Multiply", and require only simple arithmetical operations, but allow a choice of operands, when frequently there are many on each side of the equation. Another degree of freedom is which type of operation to take at any given point. This is no longer so formulaic. The sample proof lists motivations for each step, but these, again, are a matter of choice. There is the overarching goal of isolating the y variable, and every motivation along the way comes, somehow, from that goal, as well as from the bottom-up constraints of which operations lend themselves to the goal and are legal to carry out, given the form of the equation up to the current point in the proof. The non-formulaic nature of everyday as well as expert-level mathematics is noted in Polya's accounts of heuristics that can be used to help one find the next steps to take [Polya 1954]. The mention of heuristics suggests the heuristic role of distributed processes in perception as proposed in Chapter 2.

[5]And also, unknown additional imagery and intuitions.

Because picking the next step is not algorithmic, mathematicians learn that even an easy problem can turn into a bit of a search for the right path to the goal, as one frequently may take a step or two that lead to a dead end and must be erased. This phenomenon of revision is important to the second half of this chapter, but for now, the essential observation is that the novice's work also has one of the traits vital to creativity — namely, opacity (of course, the novice's work is also productive, so that it satisfies at least two of the traits of creativity). Opacity is a condition that applies to the *reasons* nehind the steps in a proof. What the novice is doing in simple mathematics is using distributed processes to choose among learned mathematical operations that are, themselves, algorithmic processes. In contrast, the Ramanujan-style genius uses distributed processes to choose from operations that are themselves, apparently, distributed processes; that second level of operations is self-taught.

It is argued here that the traits of Creativity that seem to apply to the genius's work but not to those of the novice or expert occur by correlations that follow from the picture as painted thus far. An individual who can learn and master self-taught mathematical techniques will have some techniques for mathematical work that happen to be distributed processes. Distributed processes, often quite complex owing to their vastly higher fan-in, will tend to encompass more variety among themselves than algorithmic processes. Because of this, one self-taught mathematician's techniques are apt to be different from those of another self-taught mathematician, or from those of the community at large. Therefore, three things — techniques that are self-learned, those that employ distributed processes, and those that yield a high degree of originality — are highly correlated with one another. In large part, the remaining traits of creativity that distinguish the genius from the novice follow from this. Exceptional value is rare without originality and talent is essentially synonymous with one's work having value.

What is striking is that the dichotomy between algorithmic and distributed processes essentially "invents" creativity. The mathematical operations that can be taught are the algorithmic ones — those that leave the thinker with an awareness of the related thoughts. When an operation has identifiable operands that are explicit in the input (low fan-in), and the operation can be learned, then it is algorithmic. Because these operations are taught, their use becomes common, and the results that come from them tend not to be original and thus lack value, and hence their practitioners are deemed to lack special talent. Achieving distinction thus requires having one's own private stock of self-learned, self-derived techniques. A more "conventional" genius like Gauss will learn the community's techniques first, but also forge ahead to discover new ones. In some cases, these will lead to "algorithmizable" techniques that can be taught to others and advance the community. In other cases, these will be couched only in terms of distributed processes that the gifted mathematician will *not* be able to teach to others, and will remain part of the private stock of the genius. One might say that the techniques that lead one to be deemed "creative" are distinguished less by "insight" than by their lack of "outsight" (any description that makes the technique teachable).

To summarize this section in terms of the race metaphor, one might expect the tortoise to beat the hare when the start and finish of the race allow a shortcut through the water, whereas one would expect the hare to win when the track is all on land. The different strengths of the contestants and the nature of the racecourse will play a crucial role in determining who wins the contest. "Creativity" may largely be a matter of being the only swimmer around in an area where aquatic shortcuts are plentiful.

The goal of this section was to demonstrate that the division of productive processes into distributed and algorithmic processes is a natural one. "Creativity", as a concept, is largely an outgrowth of the characteristics of these two types of process, and does not have any other natural underpinning. The mystery of creativity resides in the mystery of the mechanisms of distributed processes. What was asserted in Chapter 3 will be asserted here, as well: the challenge calls for enormous amounts of future effort, but it is not in principle insurmountable.

4.4 First-principles creation vs. review and revision

In countless domains where one seeks the highest-quality results possible, a useful strategy is to make multiple attempts and then to select the best one from them. In some endeavors, a related strategy is to improve upon an initial attempt in order to produce progressively better versions [Boden 1990; Schank and Childers 1988]. This is true in writing, art, and engineering, and it should surprise no one that it is also true in font design. The account that one professional typeface designer gives of how he works gives ample evidence of this [Abes 1994]. The strategy by which one evaluates past work and then produces new work (whether a new attempt from scratch or a modification of earlier attempts) will henceforth be called "review and revision."

In a related phenomenon, the manner in which paths through physical space are navigated makes the power of review and revision abundantly clear. Probes launched on interplanetary missions achieve remarkable accuracy in reaching intended targets. This is often compared to other tasks in which such accuracy is hard to come by. For instance, the head of the navigation team for the Mars Pathfinder lander said:

> To give you an idea of the accuracy that we have achieved here, this is the equivalent of playing a round of golf in which the hole is in Houston, Texas, and the tee-off is in Pasadena, California. We're basically hitting a hole in one here.

Frustrated golfers may take solace in three reasons why the comparison is misleading. First, space probes are directed by computers that can, of course, aim rocket engines with far more accuracy than shaky human arms can direct the motion of golf clubs. Second, motion in nearly empty interplanetary space is governed by very few forces, and they are all very well known. Third, and vital to the discussion in this section, space vehicles are redirected throughout their flight! This is not possible in golf, and if it were, a hole in one would not be that difficult. Under the normal rules, hitting a hole-in-one golf shot from California to Texas would be difficult, even for a golfer with enough strength to get the necessary distance out of the shot. However, it is not difficult to pick up a golf ball, get into a car, drive from California to Texas, and then place the ball into a given hole on a given golf course. The key is that the driver using a car may change direction countless times, as seems appropriate from the view through the windshield. Space probes, too, change direction on the way to their destination, to compensate for errors in their current heading. Thus interplanetary navigation tends, in fact, to be rather like completing a hole of golf after several strokes, with a few direction changes affording far greater accuracy than would be possible in one single long ballistic shot.

Navigation in physical space is of course different from the production of artifacts, but even so, the power of review and revision applies. Pencils have erasers, keyboards have backspace keys, and writers edit their work; all of these help improve the quality of the final product.

4.4.1 Objective evaluation, by assumption

Given some way to objectively evaluate quality, one could empirically evaluate different strategies for review and revision in artistic production. The difficulties of evaluating aesthetic quality objectively were discussed in the last chapter. Despite those thorny issues, a useful series of thought experiments might nonetheless begin with the assumption that some way of arriving at objective, quantitative quality ratings is available. Fortunately, it will not be necessary to actually produce such a function, nor even to specify what sort of art the hypothetical scenario concerns.

Suppose, then, that some hypothetical artist's work can be mathematically scored on a scale from 0 to 100. A set of works by the artist receives a set of ratings, one per work. The number expressing each rating stands for the percentile rank of that work's quality. All that is needed to simulate this is a number generator that produces a flat distribution in the given range. With the added assumption that the artist's work fluctuates randomly in quality from one product to the next, a standard pseudorandom number generator gives exactly the kind of output one would expect if this were being done with an actual evaluation of a real artist's work.

The assumption that an *objective* evaluation function exists is, of course, not quite sound. It would be enough, however, to have a *consistent* manner of evaluation. With a fixed human as the evaluator, the assumption of consistency is probably a decent approximation of reality. This can be the real-world counterpart to the scenario, and the grain of salt to take with it is the consideration that another viewer might rank things differently.

4.4.2 Thought experiments on review and revision

Genius is two percent inspiration and ninety-eight percent perspiration.

— Thomas Edison

The average rating of works that the artist in the thought experiment produces will by definition tend towards 50.0. In all the thought experiments in this section, the full artistic product will be a set of 26 smaller creations — not coincidentally, the number of gridletters in a gridfont. (No gridletters will actually be created in these thought experiments — just the ratings that might result if gridletters *were* created.)

For the sake of simplicity, the quality of the full product will be simply taken to be the average of the quality of the constituent creations. For a set of this, or any size, the artist employing no revision will thus earn an expected average rating of 50.0.

One simple strategy employing revision simulates the creation of two entire gridfonts. Then, once each gridfont has been rated as a whole, the better one will be kept and the worse one discarded. Simulation of this using a random number generator shows that this strategy yields an average quality of about 53.3. It is critical to note that for this strategy or any other revision strategy to produce a benefit, there must be a nondeterministic element to production. If identical situations always result in identical output, then all the candidates up for consideration in any circumstance will be identical, and revision will yield no benefit over the production of a single attempt.

The same amount of work (in terms of number of gridletters generated) with a smarter strategy can lead to higher quality. Instead of keeping the best entire gridfont out of two created, the artist can keep the better-rated gridletter (out of the two created) in each category, and assemble those 26 letters into a gridfont. Experimentation shows that this gives an average quality of about 66.6, much better than the revision strategy that keeps or discards an entire gridfont at a time.

Whether to treat an entire gridfont or individual gridletters as the unit to revise is one variable in choosing a strategy. Another variable is how many attempts at each category one should make. If taking the better of two is helpful, then taking the best of three or more is sure to be more helpful. Table 4.2 shows the results of simulations that try each strategy with four different levels of attempts-per-category. Each strategy was tried 1000 times, and the average rating for each condition is shown here.

It is easy to see why the strategy of making changes one gridletter at a time is superior to making changes on a gridfont-by-gridfont basis. Equally obviously, quality also increases with the number of attempts made per category.

	1	*2*	*10*	*25*
by-font	50.1	53.3	59.0	61.1
by-letter	49.1	66.6	91.2	96.6

Table 4.2: Revision strategy and number of attempts vs. quality.

The power of revision strategies is made even more vivid with simulations of artists of differing levels of talent. Talent is simulated here by altering the distribution of simulated quality; the median rating can be elevated by raising the output of the original random number generator to a power. Different powers yield different distributions, and so simulate artists of different levels of ability. As always, scores refer to the percentile rank in terms of the artist whose work has a median (and mean) score of 50.0. Table 4.3 shows how talent and number of attempts using the by-gridfont revision strategy relate to quality.

Median Score	*1*	*2*	*10*	*25*
0.5	50.8	66.9	91.3	96.60
0.75	65.8	82.7	98.3	99.62
0.9	76.9	90.3	99.47	99.86
0.99	85.9	95.7	99.65	99.87

Table 4.3: A greater number of trials can overcome superior "talent".

The results (not surprisingly) show that number of attempts and talent both lead to high-quality output. The interesting aspect of the table is to consider how the least talented artist can exceed the performance of more talented artists by application of the revision strategy (assuming that the more talented artists do not also employ the revision strategy!).[6] This relates to many real-world situations. For instance, the

[6]Taken to its extreme, an exceedingly large number of attempts might enable an essentially talentless creator to produce something of value, if a talented person or process reviewed the output and always selected from it that which was worthwhile. This calls to mind the hackneyed old claim that a million monkeys, given a million years, could type the works of Shakespeare. As empty as this strategy may seem, it will be shown later that some AI projects have utilized some elements of it. The figures offered happen to be a serious underestimate; a million monkeys in a million years would have a 50–50 chance of duplicating a given goal string of about three or four English words. The entire works of Shakespeare would require vastly more time, or monkeys, or both.

quality of writing (a medium that allows for revision) tends to exceed that of speech (which allows for only very limited revision in one's head). Hard work pays off in a number of domains.

The relevance of this to computational models of creativity is that revision, clearly, leads to better performance. Special emphasis should be given to the fact that revision can help models of creativity in two ways. It can make them more humanlike (because people review their work and revise it as needed), and it can make them produce work of higher quality.

It is also worthwhile to rephrase the comparison in terms of the weakness of never revising. The mean ratings of quality stemming from that approach appear in the column showing quality based on '1' attempt in Table 4.3. For a program that never revises to match the quality of a program that *does* revise, its first-pass attempts must be of exceptional quality. To produce better work than a program that keeps the better of two attempts at each product, a nonrevising program's per-attempt median rating must be over the 75th percentile. To match a program that keeps the best of ten attempts at each product, even a per-attempt median rating at the 99th percentile is not enough.

4.5 Proficiency of a modular program

...their appearance and their work was as it were a wheel in the middle of a wheel.

— Ezekiel 1:6

This section offers an intuitive way to understand the proficiency of a model implementing a feedback loop of creativity by means of discrete modules for review and revision, respectively.

A useful concept is that of a schematic diagram that geometrically represents prototypicality of the members of a category (graded prototypicality is discussed in

[Lakoff 1987]). Points close to the center indicate objects that are strong members of the category, and increasing distance from the center indicates decreasing prototypicality. Such diagrams resemble Venn diagrams, but with a sharply-delineated boundary that indicates just one of infinitely many levels in quality, where quality as a member of the category is graded, and increasing towards the center.

To make such a diagram requires only a set of objects and some metric of quality. Earlier publications on Letter Spirit used such diagrams to show the quality of certain gridletters as 'a's [McGraw 1995]. Figure 4.3 shows twenty-three gridletters arranged according to how well they represent the abstract idea of 'a'-ness. The best 'a's are located near the middle, with more exotic (and harder-to-recognize) 'a's toward the outside. It should be noted that all the diagrams in this section are based on approximate and intuitive measures of quality, with no great care taken to actually calculate quality in any principled way or to locate the points in the diagrams precisely. It would be possible to arrange a diagram like Figure 4.3 according to any of a number of measures of 'a'-ness. The results of experiments measuring how often humans identified given gridletters would be one way. The same approach using frequency of identification as 'a' by a given computer program would be another. Quality assessments produced by humans or by a given computer program would offer yet more ways to do this. As was noted earlier, the diagrams here are not based on any one formal measure of quality but merely on rough intuition.

A second example, in Figure 4.4, displays gridletters for their Boat-ness, or their quality as members of the gridfont Boat. The style-recognizing Adjudicator module of Letter Spirit, if trained with seed gridletters from Boat, could serve as the measure of quality (or, again, human judgments could do the same).

A bit more abstract, Figure 4.5 shows the (abbreviated) names of gridfonts located according to the overall letter-category recognizability of all 26 of their gridletters, the gridfonts closest to the center being those whose constituent gridletters are easily

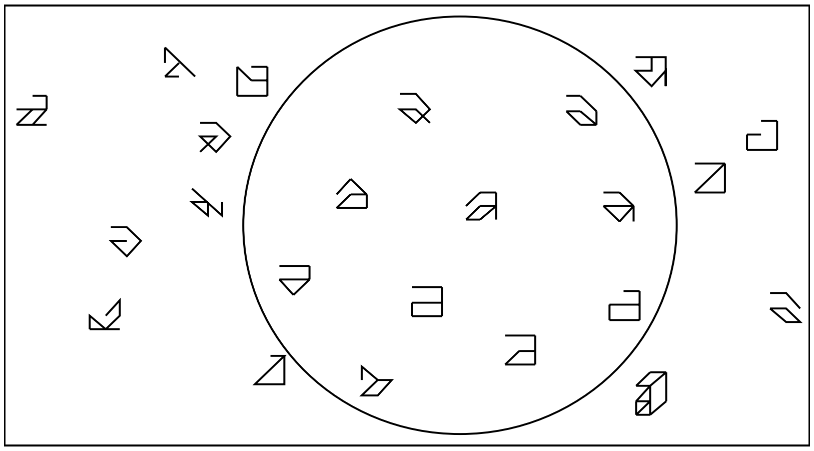
Figure 4.3: Gridletters arranged by letter-category recognizability.

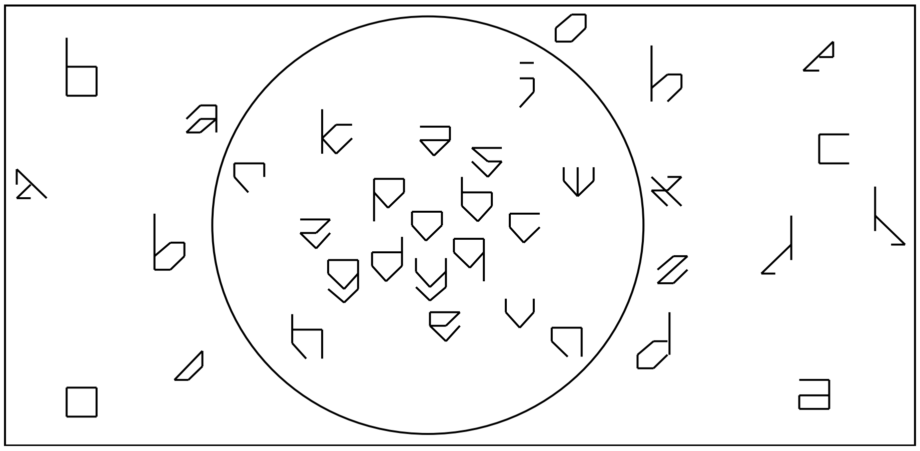
Figure 4.4: Gridletters arranged by Boat-ness.

identified as members of their intended letter categories. Figure 4.6 takes the approach a different direction, by arranging the same gridfonts according to the stylistic coherence of their gridletters. Note that here the discrepancy between quality as rated by a letter-savvy human and quality as rated by a letter-amateur or by the Adjudicator will be important. The designer of these gridfonts will often have found an abstract coherence that eludes the naive viewer.

As with the usual kind of Venn diagrams, the real point of making this kind of diagram is to illustrate interactions between categories. With three diagrams, one

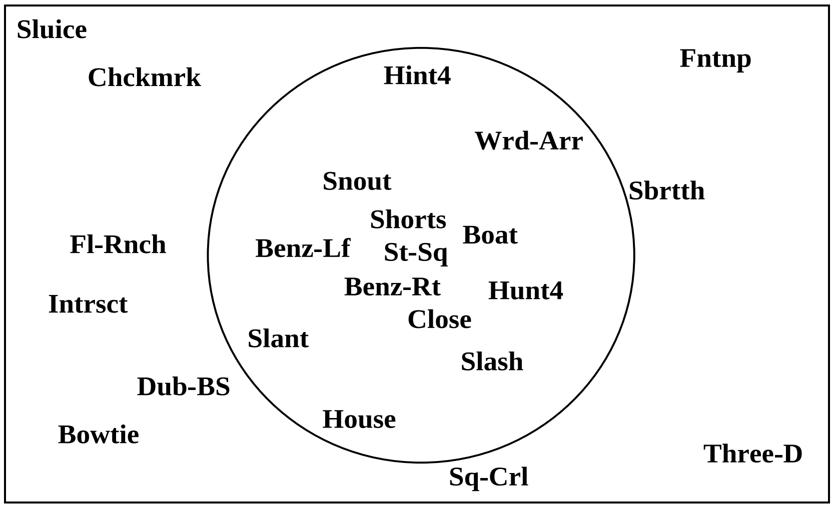

Figure 4.5: Gridfonts arranged by letter-category recognizability.

for each Letter Spirit module, the diagram is divided into eight areas, one for each possible permutation of Examiner/Adjudicator/Drafter proficiency. The modules' proficiency with respect to a given gridfont can be tested, and the gridfont located on a diagram like Figure 4.7.

Producing a diagram of this kind, which locates the 23 example gridfonts appropriately, is one goal of this thesis. The finished figure[7] (Figure 4.7) is an interesting depiction of the interrelated nature of the three modules' tasks, with respect to a variety of possible styles.

Moreover, the diagram will also facilitate the ultimate analysis of the revision strategy as implemented in Letter Spirit. Letter Spirit performance on a gridfont that falls outside the expertise of one or another of the three modules is certain to be subpar; any such failure says nothing about the overall strategy, and may indicate a mere conceptual limitation built into one module. However, for those gridfonts upon which all three modules perform well, the performance of the entire Letter Spirit

[7]Of course, any sharp boundaries must be determined by arbitrary thresholds of proficiency. The exact trio of boundaries will be chosen so that each falls at a threshold that seems to divides the example gridfonts reasonably.

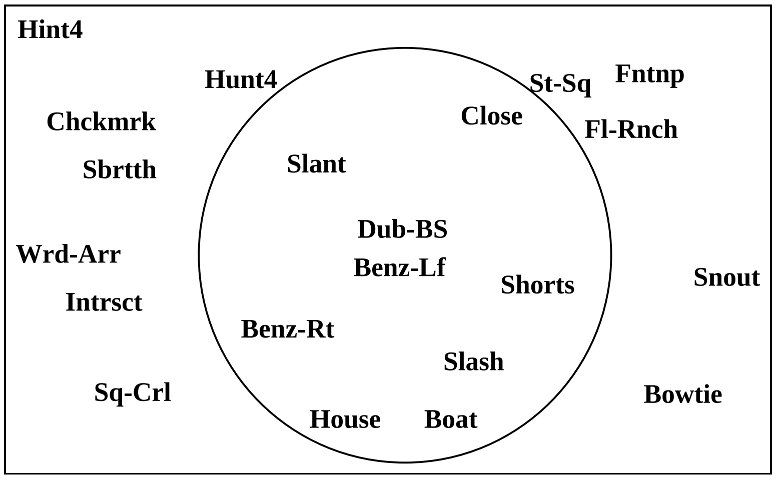

Figure 4.6: Gridfonts arranged by style recognizability.

program is a true test of the strategy of integrating the three modules together into a whole. As it turns out, six of the 23 example gridfonts fall into this category, the central area of Figure 4.7. Consideration (in Chapter 8) of how Letter Spirit performs on these six gridfonts is therefore of utmost interest.

4.6 Conclusion: Theory and practice

4.6.1 Goals for a model of "creativity"

The discussion in this chapter touches on the Letter Spirit project in a few ways. First, it has aimed to clarify the notion of "creativity", with special emphasis laid on creativity's underlying aspects of production and high quality. Second, it extends to productive processes the distributed–algorithmic distinction that was first identified in perceptual processes. How the Drafter models distributed processes for production is discussed in Chapter 7. Finally, it describes a number of phenomena related to revision strategies of aesthetic production.

Working on the assumption that creativity is a coherent cognitive phenomenon,

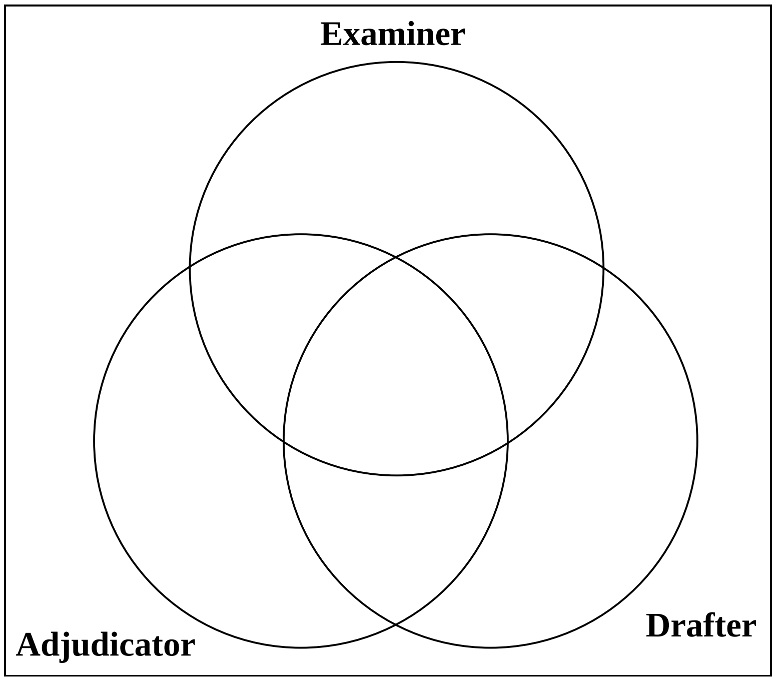

Figure 4.7: A way of representing performance by gridfont and by module.

a research project would try to define the phenomenon and to incorporate all of the characteristics in the definition into a computational model. One such candidate definition comes from earlier writings on Letter Spirit [Hofstadter and FARG 1995]. That definition was, in fact, couched in terms of properties of a computer model:

- the program itself must arguably *make its own decisions* rather than simply carrying out a set of design decisions all of which have already been made, directly or indirectly, by a human;
- the program's knowledge must be rich — that is, each concept must on its own be a nontrivial representation of some category with flexible criteria for judging degrees of membership, and among diverse concepts there must be multiple explicit connections;
- the program's concepts and their interrelations must not be static, but must be

flexible and context-dependent;

- the program must experiment and explore at a deep conceptual level rather than at a shallow surface level;
- the program must be able to perceive and judge its own tentative output and be able to accept it, reject it, or come up with plausible ideas for improving it;
- the program must gradually converge on a satisfactory solution through a continual process in which suggestions coming from one part of the system and judgments coming from another part are continually interleaved.

Another account, contrasting slightly with Hofstadter's, is found in [Johnson-Laird 1988], holding that creativity results in products meeting the following properties:

- they are novel for the individual who creates them;
- they reflect the individual's freedom of choice and accordingly are not constructed by rote or calculation, but by a nondeterministic process;
- the choice is made from among options that are specified by criteria.

This chapter rejects the idea that creativity is a phenomenon grounded in any particular mechanism, except that certain traits usually associated with the term "creativity" tend to co-occur. The correlation of these properties is due to the nature of underlying cognitive mechanisms, rather than being the direct result of a distinct mechanism that exclusively turns out creative output.

The conclusion that there is no special mechanism for creativity means that a model of creativity should aim to model mechanisms that are part of human cognition and should pursue strategies that maximize the positive qualities often associated with creativity. A model that pursues this goal by means of a revision strategy should thus incorporate the following traits:

- the program produces some product;
- decisions leading to the product are made in an informed way;
- many of the decisions leading to the product are nondeterministic;
- the program evaluates its own output and has the ability to revise output deemed to be of low quality.

These four traits succinctly meet most of the requirements put forth by Hofstadter and Johnson-Laird, fulfill most of the traits listed in Table 4.1 as definitive of creativity, and, as subsequent chapters will make clear, are incorporated into the Letter Spirit program. Some of the requirements suggested elsewhere are matters of degree — for example, Hofstadter's requirements of richness and flexibility. It is not a simple matter of whether or not a program exhibits these traits or not, but to what degree it exhibits them. Letter Spirit incorporates them reasonably well, as is documented in later chapters.

The only two traits in Table 4.1 that may not hold true for Letter Spirit are opacity and mystery. Opacity is perhaps modeled in the sense that certain types of data created by the program are open to access by other parts of the program, while others are not. Obviously, all data structures in the program are visible to some part of the program or other. Only if the program were intended to be a serious model of consciousness would a strict accounting have to be made regarding which kinds of information should be available to which parts of the program, but the project does not at present pursue this goal. Finally, mystery is the one trait from Table 4.1 that ought to be rejected as being even correlated with the others. Although high-quality productive behavior is difficult to understand, this work proceeds under the assumption that such understanding is not, in principle, impossible.

4.6.2 AI survey

Up to this time, computational models of cognition have tended not to incorporate humanlike review-and-revision strategies. The survey of AI models of creativity in [McGraw 1995] finds that researchers usually try to create a program that generates relatively high-quality output on its first try. This is in spite of the fact that several researchers have found it useful to *personally* sift through their data by hand and present to the world only the best data that their program creates (with "best" meaning best in the eyes of the researcher). At least one exploration into creativity involved the fruits of the researcher's pruning of the program's input being fed back into the program as part of its routine [Lenat 1982]. This certainly achieves respectable quality of output, but the extent to which the output can be said to be the *computer's* work is seriously compromised. It is particularly curious that an effort to model creativity can rely upon human revision (done by a human, outside the model) without provoking a serious interest in modeling the process of revision.

In a very narrow and literal sense, review and revision has been part of computer algorithms for a long time. Generate-and-test algorithms — those that perform some sort of test before committing to some output or course of action — are ubiquitous. For example, a program that attempted to break into a computer account might try a known username with every word in the dictionary as a candidate password. It will easily find many passwords, but such brute-force algorithms strike no one as creative.

One innovation in Letter Spirit is that a sophisticated process of review is utilized as part of a sophisticated process of revision. The best previous models of creativity have tended to focus on sophisticated means of achieving high quality in the output in a first pass, with either no checks at all, or only primitive and formulaic checks at best, applied to the output before it is delivered in its final form to the outside world.

For example, the humor-creating program JAPE produces punning riddles [Binsted 1994]. (A sample of JAPE's work: "What kind of pig can you ignore at a party? A wild boar.") The creation of such output follows one pass that combines a definition of what a pun is with a lexicon, in the hopes of generating novel puns. This output is pruned, a bit, by two formulaic checks that look for circumstances that always ruin the joke (one of them screens to see if the words used in the question and the answer happen to be the same; the other weeds out cases where the answer is a genuine common English phrase). While a small number of simple checks like this do prune the output and raise the quality from the pre-check stage, such a revision process is fixed and extremely limited. Although the processing might be described as taking place in stages (a small number of stages), it is still a straight-line program. This could just as well be considered a first-pass approach with no revision, with the checks considered part of the initial production requirements. In fact, the author of JAPE has interpreted the program more as a *definition* of a certain style of joke than as a *model* of the processes by which people create such jokes [Kim Binsted, personal communication].

Some models apply review and then revision in an ongoing process, and this has been true of FARG models before Letter Spirit [Mitchell 1993]. When Copycat runs, it evaluates its representations repeatedly in the calculation of temperature, very much like that in Letter Spirit's Examiner, which was inspired in large part by Copycat. Temperature, in Copycat, assesses the quality of the representation that has been built up during a run. The calculation is essentially the weighted sum (with fixed weights) of two weighted sums (many of the terms of which change throughout the run). A similar formula is used for the Examiner's calculation of temperature. In both programs, temperature can influence the chances that subsequent changes will be made to the representations that have built up thus far. The halting conditions in Copycat and in the Examiner, which are influenced by temperature and activation, are also a kind of review, but one that is boolean, registering approval of the prospective answer only once, at the end of the run.

A very different kind of model of creativity — the genetic algorithm — inherently depends upon some sort of review process, which is used to select those members of a population that will survive into the next generation. Because the specifics vary considerably from one implementation to another, it is difficult to make many generalizations about genetic algorithms models except to note that they all employ review and revision in some fashion or other.

In yet another branch of cognitive modeling, many case-based reasoning (CBR) models employ review and revision. The CBR system CLAVIER [Hennessy and Hinkle 1992] finds ways that aircraft parts, during the production process, can be loaded into an autoclave (a sort of industrial oven that treats metal to increase its strength). As CLAVIER adapts arrangements of items that have previously been found to work, it tests the new, prospective arrangements to see if they will fit inside the autoclave.

For a final example, the program Phineas [Falkenhainer 1990], an extension of the Structure Mapping Engine architecture (see [Falkenhainer et al. 1989]), uses analogy to find physical theories to explain novel behaviors that are presented to it. Like Letter Spirit, Phineas is based on modules that themselves are relatively complex, some involved in generating candidate theories, some involved in testing candidate theories. Its basic operation employs what Falkenhainer calls the *map/analyze cycle*. In this cycle, the first step is to try to explain a behavior in terms of theories already known to the system. If this cannot be done, then an analogy-producing module tries to map the facts of the new behavior to known situations in order to produce a new theory that may explain the behavior. Another module is then used to test the proposed theory to see if it explains the behavior in terms of previously-held background knowledge in a way that can be tested experimentally. If not, a module that reasons with background knowledge looks for inferences that may extend the analogy further. This cycle is carried out repeatedly as the program looks for a theory that can be tested experimentally.

These examples show that revision strategies are not new, but that there are a variety of ways that they may be implemented, some more sophisticated than others. Most basic are those programs that simply do not employ review or revision in any way. Next are the programs that review their work by one or more checks, but only after all generation of candidate output has been produced. In these cases, the review is a mere pruning of the output, and the flow of control is still basically that of a straight-line program, which does not benefit from the power of a full revision strategy.

By contrast, a program like Phineas employs iteration with progressive refinement of its initial attempt to explain a situation. The revision strategy in Phineas takes a structured candidate output (a theory) and, if necessary, augments it (with reasoning and analogy), and tests it to see if it is acceptable (that is, if it produces a testable explanation of the situation). Modifying the structure of an initial candidate output is also the revision strategy of Copycat and of the Examiner. In all three cases, a single change to a part of the structure may drastically affect the quality of the whole structure, for better or for worse.

None of these strategies is quite like that in the simulation of revision earlier in this chapter, in which the candidate output is a whole that consists of a set of distinct elements with a fixed size. Unlike the structured representations of Phineas (a theory), of Copycat (an analogy to a transformation from one string to another), and of the Examiner (a parsing of a gridletter into parts), the simulation's representation allows for the modular replacement of any element. The elements are rated independently of one another, and the rating for the whole is the mean of the ratings of the parts. This is, however, approximately the situation with Letter Spirit — with some caveats. For one, changing individual gridletters should change the sense of style for the gridfont, and therefore, the rating. However, changing one gridletter should not change the style of a gridfont very much, once several gridletters have been established as members of the gridfont. A second caveat is that it is not clear that, in any gridfont, the mean of the parts' ratings is the right way to get at the whole's rating.

In summary, models of creativity that employ review and revision strategies differ in the dynamics of what revision takes place as a result of review. They also differ in the complexity of the review function. The password-cracking program has a trivial evaluation function (success or failure). JAPE has a pair of simple formulas. The calculations of temperature in FARG models are useful and not necessarily trivial, but are considerably simpler than the review process in Letter Spirit, which is based upon the Examiner and the Adjudicator. That the sophistication of Letter Spirit as a whole is greater than that of the Examiner alone is especially clear, because the Examiner is just one part of Letter Spirit! No attempt will be made here to appraise the precise degree of sophistication of each model's review process, and to claim that Letter Spirit's is the best of the bunch, but it is worthwhile to note the considerable variety along these lines. The richness of Letter Spirit's review process is among the project's best attributes, and it emphasizes the importance of the revision process, as opposed to putting most or all effort into making a model produce its best work in a one-pass approach.[8]

Beyond sheer sophistication, Letter Spirit's review process is apparently unique in striving to use a complex model of aesthetic perception as the basis for revision. All other examples seen so far (and in the table below) either use a simple function as a filter for aesthetic quality (e.g., whether or not a prospective pun reinvents a

[8]The programmer-years put into development of the various Letter Spirit modules provides a rough gauge of the importance of review. The development of the two modules responsible for review took up over three-quarters of the time spent on the project's development effort to date. Chapter 10 elaborates on the development of the project.

preexisting term in the language) or use a function that evaluates the prospective output with a boolean determination of success or failure (e.g., whether or not the theory explains the situation) or base the evaluation on a matter of (simulated) physical reality (e.g., whether or not a set of aircraft parts fits into an autoclave). One last way to evaluate the output is for the researcher to do it; this seems to have been the case for many published works. If the purpose is to contrast the kind of work that the human–computer hybrid system produces with that of a human alone or a computer system alone, this might have some scholarly interest.[9]

One final essential factor that distinguishes various approaches to review-and-revision strategies is how they introduce variety into their candidate answers. For any model of creativity that uses a review-and-revision strategy to work, any initial candidate answer that is proposed must, if rejected, be replaced by a different candidate answer. Proposing and rejecting the same candidate over and over accomplishes nothing. If the number of possible outputs is very limited, trying all of the possibilities may be feasible, but this is only the case for very few domains, and not ones liable to be associated with creativity. The alternative put forth in the simulation, and the one that Letter Spirit and many other models of creativity employ, is nondeterminism, in which successive attempts at producing a candidate produce several distinct versions, and the revision strategy selects whichever one that is rated best. Another approach would be to have a method of producing candidates by running down a list in some canonical deterministic order.

Table 4.4 shows how a number of AI models of creativity compare with regard to the criteria for creativity listed earlier in the conclusion. In most cases, the criteria that allow for beneficial use of a revision strategy are lacking; in those cases, quality

[9]It seems that the goal, in some if not all such cases, has been to give the program a boost and thereby help it generate better output. If this were the point, having the human do all the work might be the best way to cut to the chase.

is constrained by the work the program can do in a first pass. Many of these are discussed in greater detail in [McGraw 1995]. Some have been mentioned earlier in this chapter, while others appear only in the table [Chamberlain 1984; Meehan 1976; Lenat 1983; Langley et al. 1987; McCorduck 1991; Cope 1991].

Program	*Productive?*	*Informed?*	*Nondeterministic?*	*Revision?*
Racter	yes	yes	yes	no
Tale-Spin	yes	yes	no	no
AM/Eurisko	yes	yes	yes?	limited
BACON	yes	yes	no	no
Jazz-Improviser	yes	yes	yes	no
Aaron	yes	yes	yes	no
Copycat	yes	yes	yes	limited
EMI	yes	yes	no	no
JAPE	yes	yes	yes?	no
Phineas	yes	yes	?	yes
CLAVIER	yes	yes	no	limited
Letter Spirit	yes	yes	yes	yes

Table 4.4: Models of creativity have tended to focus on first-pass approaches.

Letter Spirit is probably the only model of aesthetic creation that uses a reasonably rich model of aesthetic perception to drive a revision strategy. That being said, there are many fine models of creative behavior that preceded Letter Spirit, and many have their own positive attributes apart from the criteria listed in the table. For example, Phineas, Copycat, and the Examiner module have the ability to *modify* previous candidate outputs at various levels throughout the candidate's structure, while Letter Spirit can only review and replace on one level — that of the gridletter, the part that lies one level beneath the whole. Taking the lead set by Copycat, Phineas, and (ironically) Letter Spirit's own Examiner module, an attempt to endow Letter Spirit with the ability to modify its output on different levels, especially those below the letter level, would be an important direction for future implementations. For now, a basic revision strategy is in place. With the discussion of the theory behind Letter Spirit at an end, it is time to describe its implementation.

Part II

Implementation

CHAPTER FIVE

Examiner

5.1 Introduction

The Examiner is the Letter Spirit module that determines the letter category of an input gridletter. It was implemented by Gary McGraw for his doctoral thesis, and this chapter cannot possibly rival the thorough description given there [McGraw 1995]. It is necessary, however, to discuss it here for two reasons. First, the Examiner has undergone significant revision since 1995, and this chapter describes the new version, focusing on the changes from the 1995 version and the reasons for them. Second, having a chapter on the Examiner means that this document provides, under one cover, a parallel treatment of all three Letter Spirit modules.

5.2 Implementation

5.2.1 An overview of processing in the Examiner

The Examiner works towards the goal of finding a parsing of the input gridletter into parts that are appropriate role-fillers for a given role-set. The letter category that that role-set represents is returned as the answer; the way that the gridletter was parsed is also useful information, and is used by other portions of the Letter Spirit program.

Like other FARG models, the Examiner models cognition on a fine-grained level. Its activity can be considered either on that level or on a higher level, which emerges from the nondeterministic action buzzing about on the low level. In terms of the high level, processing is quite unlike that of a conventional algorithm. An outside observer might perceive the program's activity as occurring in steps, but the steps are more like trends, often gradual, in the program's buildup of a representation. Some of the high-level steps logically precede other ones, but in many cases, two or more high-level steps occur simultaneously, reflecting the fact that the low-level operations that underlie them are interleaved. The preceding general comments apply, in many ways, to all three Letter Spirit modules.

Processing in the Examiner goes through the following series of high-level steps to achieve successful recognition of the gridletter as a member of a letter category:

1. Based purely on superficially evident high-level properties of the gridletter, some letter categories are rapidly noted as being more likely to be the final answer than others. This information is used both to boost the activation of the relevant role-set nodes in the Conceptual Network, and also to segment the gridletter.

2. The gridletter is segmented into plausible parts. The verb "segment" is used to refer to the division of the gridletter into parts; "parse" is used to refer to segmentation followed by subsequent identification of the resulting parts in terms of the roles they fill. Figure 5.4 shows how a gridletter has been segmented and parsed (this example is explained in greater detail shortly).

3. The parts are given semantic labels such as "tall", "skinny", etc.

4. The labels attached to a part are compared to the norms for roles, and if there is a close match between the labels attached to a part and the norms of a role,

a process known as "sparking" associates the role and the part in two ways: the role is bound to the part, and the role's concept in memory receives positive activation.

5. Activation in memory spreads, so that active role concepts lend activation to related role-sets, and vice versa. (A more detailed explanation follows later.)

6. If the activation of one role-set is much higher than that of any other role-set, then the program quits, returning that role-set's corresponding letter category as the answer.

In many runs, things will be much more complicated than this. Some steps, especially Step 3, are carried out by the action of many codelets, and thus the run will deviate from the sequence presented in the above list in one way or another. For example, one part might be entirely labeled and sparked before another part receives its first label. In typical runs, the high-level steps of labeling, sparking, and spreading of activation all begin before any of them end, so that the high-level steps can be said to take place in parallel.

Runs can also deviate from the sequence presented above when there is no winning role-set after every part has been labeled and has undergone sparking. Any of a number of checks may detect this, and probabilistically order a resegmentation. This essentially sets the program back to Step 3 to try that and the subsequent steps again. Easy gridletters usually require few passes through the steps (most often, only one); difficult gridletters may require many attempts to segment them in a way that results in recognition. Truly difficult gridletters will never activate any role-set. In these cases, the program eventually quits without offering an answer.

5.2.2 Memory structures in the Examiner

The Examiner uses five memory structures: the Conceptual Memory, the Conceptual Network, the Workspace, the Coderack, and temperature. All but the Conceptual Network are used by other modules as well. This section explains the Examiner's memory structures, many of which are also used by one or both of the other two Letter Spirit modules.

Workspace

The Workspace stores most of the information needed to capture the state of an Examiner run in progress, and as such, its contents change very frequently throughout the course of a run. It consists primarily of a list of parts into which the input gridletter has been segmented. Each part is stored as a list of the quanta of which it is comprised, and the labels, if any, that have been attached to it. When a part has first been created, and has not yet had any labels describing the part's attributes attached to it, it is marked as "whiny". When (if ever) a part is used to *spark* (activate) roles, then information noting that those roles are *bound* to the part is added to the Workspace.

Conceptual Memory: Roles and role-sets

The Conceptual Memory is the long-term memory of Letter Spirit. It is divided into several subareas, not all of which are used by all modules. For the purposes of the Examiner, the Conceptual Memory consists of the roles that underlie letters, and the role-sets that describe how letters are composed of those roles.

Roles in the Conceptual Memory

Ultimately, all three modules of Letter Spirit need to have some specific knowledge about the letters of our alphabet, and this knowledge is provided by the Conceptual Memory. The Conceptual Memory is a long-term store that defines properties of entities in the three levels of Letter Spirit's letter-representation hierarchy: roles, role-sets, and letter categories. The Conceptual Memory in the current implementation of the Examiner differs significantly from that of McGraw's Examiner.

The Conceptual Memory defines each role with a set of norms — the values of such qualities as height, width, and curvature that are expected for a part that fills that role. The norm definition of a role does not specify just one allowable value per dimension (for example, declaring "tall" as the only allowable height for a "left-post"). Rather, the norm for a role in any given dimension is implemented as a weighted list of possible values. To be classified as a role-filler for a given role, a part need not match the highest-scored values of each dimension in the role's definition; the overall quality of the match between the part and the role is calculated by taking each property which has been attached to the part as a label, finding the weight that the role's definition associates with that label, and summing all of those weights to produce a graded membership score for the part's appropriateness as a role-filler for that role.

The full details of how one role, "left-post", is defined in terms of norms is offered below. Role definitions of this kind are used not only by the Examiner, but also by the other two Letter Spirit modules, and these definitions are thus very much at the heart of Letter Spirit. This information is fairly dry and arduous to read through, and is placed here solely as a resource for the deeply interested reader. The general nature of the approach should be understood from skimming the description below:

topology	segment
stroke	down
shape	simple 40, bactrian 10, cupped 30, spiky-closure -10
neighborhood	-1 -1 0
	-1 0
	0 0 +1
contact	b1: middle-and-bottomtip 20, middle 11, bottomtip 11, middletwice 18
	b2: middle 20, bottomtip 15
	h1: middle 20, bottomtip 11
	k1: middle 20, bottomtip 15, middletwice 10
tip-1	*location*: *1* 10, *8* 8, *2* 8, *9* 6, *15*
	orientation: north 10, west 6, northeast 7, northwest 7, southeast 6
tip-2	*location*: *3* 10, *5* 7, *4* 7, *12* 6
	orientation: south 10, southwest 6, west 7, northeast 6
end-1	20-north 40, 38-southwest 13, 4-west 13, 48-northwest 27, 26-north 13, 23-north 27, 40-northeast 13
end-2	29-south 40, 30-south 10, 23-south 10, 42-southwest 10, 12-west 10, 10-west 10, 26-south 10, 42-northeast 10
curvature	straight 10, square-left 7, slight-left 7, slight-right 6, square-right 6
height	tall 10, short 8, medium-ht 8
width	skinny 10, half-wide 9, wide 6
weight	medium-wt 10, light 8, heavy 6
roof	top 10, t-height 7
floor	baseline 10, midline 8, x-height 8
left-edge	left 10, middle 6
right-edge	left 10, middle 9, right 6

Table 5.1: How left-post is stored in the Conceptual Memory.

Topology gives one of four possibilities for the topology of the role's prototype: *segment* (two endpoints, and uniform curvature throughout); *bisegment* (two endpoints, with a switchover in the middle from one curvature to a distinctly different curvature, as is the case with 'f-post'); *loop* (a closed loop); and *dot* (only the 'dot' role, used in 'i1' and 'j1', has this topology). Topology often dictates which other kinds of information need to be contained in a role's description; for example, loops normally have no endpoints, so none are given as norms.

Stroke disambiguates which of two tips found in a role-filler should be considered tip-1 and which tip-2 by indicating the general direction that leads from tip-1 to tip-2.

Shape distinguishes segments that may have unusual kinkiness and self-crossing from those that cannot. For example, neither wing of a 'v' can be *bactrian* (that is to say humped — having two changes in vertical direction), or the gridletter would actually be a better 'w' than 'v'. Other shapes are *cupped* (one change in vertical direction, like a 'U' or upside-down 'U') and *spiky-closure* (a part whose path crosses itself). Any shape that does not fall into one of those three categories is called *simple.* Most roles are free to have any of several values; the primary purpose of the shape property is to veto certain bad combinations, such as the example of humped parts within a possible 'v'.

Neighborhood is a major innovation since the 1995 Examiner. This provides information regarding where the material in the gridletter *besides* the part in question is situated with respect to that part. A 3×3 array has entries in each cell except the center one, indicating whether there should be material in a certain direction (+1), or not (-1), or whether it does not matter (0). This distinguishes between pairs of roles that are similar in terms of most other properties, such as the crossbar of 'f' and 't', as opposed to the z-cap of 'z'. In general, these two roles have very similar norms, but a crossbar will tend to have neighboring material both above and below it, while a z-cap will tend to have neighboring material below it, but not above. The 1995 Examiner used one role for what is now crossbar and z-cap, whereas the neighborhood property makes the two very distinct, and not interchangeable.

Contact is a similar innovation since 1995. To use the same example, a crossbar may be touched in any of a number of ways, usually being crossed roughly at its middle, whereas a z-cap may also be touched in many different ways, but usually only at its right tip. Ignoring how a part relates to the material around it misses an important clue as to its role-identity (or, given a bad segmentation of the gridletter, its lack thereof). Some roles have different contact norms for their uses in different role-sets, and 'left-post' is just such an example, as the four lines marked 'b1', 'b2', 'h1', and 'k1' indicate. As an example that should make the meaning of all the other entries clear, for the row 'b1', the entry "middle-and-bottomtip 20" indicates that for 'b1' left-posts, a weight of 20 is assigned to parts that are touched in two places — the bottom tip and somewhere in the middle (that is, anywhere between the two tips). Neighborhood and contact are used to prune the Examiner's search options and thus to accelerate finding the correct answer. Contact is expressed, for segments and bisegments, in terms of how the tips and the middle portions of a role-filler should be touched; for the other topologies, the possibilities are simple touching or non-touching.

Tip-1, *tip-2*, *end-1*, and *end-2* are properties that partially replace the "squares" norm of the 1995 Examiner. Rather than identify those squares that a part filling a particular role would tend to occupy, the current Examiner provides norms for the locations and orientations of a role's tips. (The quantum numbers refer to the quanta numbering system shown in Figure 1.2.) Tips and ends form an interesting contrast: *tips* lists the location and orientation of an endpoint as two separate norms, while *ends* lists norms for a bound pair of endpoint location and endpoint orientation. Both ways have their benefits and so both are used here, although they may seem to be redundant. The former allows generalization while the latter is better for favoring or disallowing specific combinations of values. This representation conundrum has often arisen for cognitive modelers, and here it seemed best simply to use both. This issue is discussed more in Chapter 10.

Curvature is a simple spectrum ranging from *strongly curved to the left*, through *straight*, to *strongly curved to the right*. Segments have one curvature norm, bisegments have two (one for each half of the role), and loops and dots have none.

Height, *width*, and *weight* (the total number of quanta) are basic norm dimensions that require little explanation. Each possible value specifies a narrow range of heights, widths, or weights.

Roof, *floor*, *right-edge*, and *left-edge* are, along with tips and ends, the grid-location-specific norms for each role. These six norms help replace the 1995 Examiner's "squares" norm, which specified location in a manner that, though useful for recognizing an input gridletter, was not much of a guideline for drawing one, nor for recognizing quirks that amount to an element of style.

Role-sets in the Conceptual Memory

Most of the detail in the Conceptual Memory is at the level of roles; still, there are two higher levels that are crucial but are far more easily defined. It is the highest level — letter category — that has one entry for each of the 26 lowercase roman letters. Each letter category is represented as a small (of size one or two) set of role-sets. For example, the 'b' letter category has one role-set designated 'b1' and another designated 'b2'. Role-sets are basically just lists of constituent roles; the norms for how the roles in a role-set should be assembled into a letterform reside in the contact and neighborhood properties of the relevant roles. For the example of 'b', the 'b1' role-set consists of the roles 'left-post' and 'right-bowl', while the two roles in 'b2' are 'left-post' and 'circle'.

R-roles in the Conceptual Memory

With letters, as with so many other things, the whole is greater than the sum of its parts. Just as a messy pile of clock parts does not make a clock, the correct role-fillers, if not arranged properly, do not make a good rendition of a role-set. Relational roles, or r-roles, are Letter Spirit's representation of how role-fillers should be put together to make the whole. When activation is spread (see below), each role-set is evaluated for how well its r-roles are filled; this evaluation is then used to lower the activation of any role-sets that score poorly. If any one of a role-set's r-roles is seriously violated, then the role-set can informally be considered to have failed the test, and it is extremely unlikely that the Examiner will call that role-set its answer. One should note, though, that r-role violation is contingent upon how the gridletter is segmented. A role-set may fail (receive an extremely low score for) one or more r-role tests early in a run, but pass the tests later, depending upon how segmentation and sparking have proceeded.

Letter Spirit has three kinds of r-roles. The *contact* r-role evaluates the extent to which role-fillers touch each other in the way that a role-set calls for. In each role's norms, each relevant contact description (for example: "touched in the middle") has a certain score associated with it. A role-set's contact r-role score is calculated by first formulating a contact score for each role in the role-set as follows: if a role is not bound to any part, then it receives a low, failing score; if, instead, the norm is bound to a part, then the way in which the part is touched is looked up in the corresponding role's norms, and that is used as the role's contact score. Finally, when each role's contact score has been calculated, the role-set's contact score is computed as the lowest (i.e., worst) of the scores that was given to its constituent roles. Essentially, a role-set passes this test if all of its roles have been bound to parts in the current segmentation and if those parts are touched by other parts in a way that suits their respective roles' contact norms; otherwise, the role-set fails the test.

A role-set passes the *filled* r-role test if and only if all of the role-set's constituent roles have been filled — that is, if each of the role-set's roles has been sparked by a part in the current segmentation. Otherwise, the role-set fails the test.

Finally, a role-set passes the *covered* r-role test if and only if its role-fillers (as specified in the "filled" test) account for every quantum in the gridletter, leaving not one left over. Otherwise, the role-set fails the test.

These three tests ensure that the Examiner returns an answer only if the parts that make up that answer stand in the proper relation to each other as well as to the entire gridletter. Later discussion of Examiner tests shows that although they may be overly restrictive in some cases, they are a good approximation of the way that a letter recognizer employing decomposition into parts needs to acknowledge that the whole is more than the sum of its parts.

Conceptual Network

The Conceptual Network is described in [McGraw 1995] as part of the Conceptual Memory, but it seems worthwhile to distinguish it here with a separate name. The Conceptual Memory, as described above, is an immutable long-term store of information on prototypes for the lowercase roman letters.

The Conceptual Network is a kind of short-term memory in the form of a localist connectionist network in which activation (between -100 and 100) expresses the extent to which roles and role-sets have been found or can be expected to be found in the current segmentation of the input. There is a node for each role and each role-set, and the nodes are connected according to the part–whole relationship; there are links between each role-set and its constituent roles. Activation spreads via these links in both the bottom-up and top-down directions. This is much as in the 1995 Examiner, although the precise roles and role-sets used now are different (being exactly the same ones used in the Conceptual Memory), and the rules governing the spread of activation are considerably changed.

One can picture the entire network as consisting of two levels: role and role-set. Links exist only between levels — there is no lateral spread of activation (lateral inhibition is a common technique in cognitive models, and has been shown to occur in the nervous system, but is not employed here). The division into two levels is explicit, as the rules governing how activation spreads differ for the two directions.

Figure 5.1 shows a portion of the Conceptual Network. The full Conceptual Network includes a node for each role and each role-set, and has a bidirectional connection between every role and the role-sets it belongs to. When activation is spread, each role-set acquires a new activation that is a fraction of its previous level plus the weighted sum of the previous activations of its constituent roles. A role's new activation, in contrast, is calculated as a fraction of its own previous activation, plus a weight times the *highest* previous activation of any of the role-sets in which it is involved. (The rationale behind the difference is explained in the subsequent section on the post-1995 enhancements made to the Examiner.) Positive activation can also be introduced to the Conceptual Network by gestalt codelets and by sparking codelets, as will be explained below.

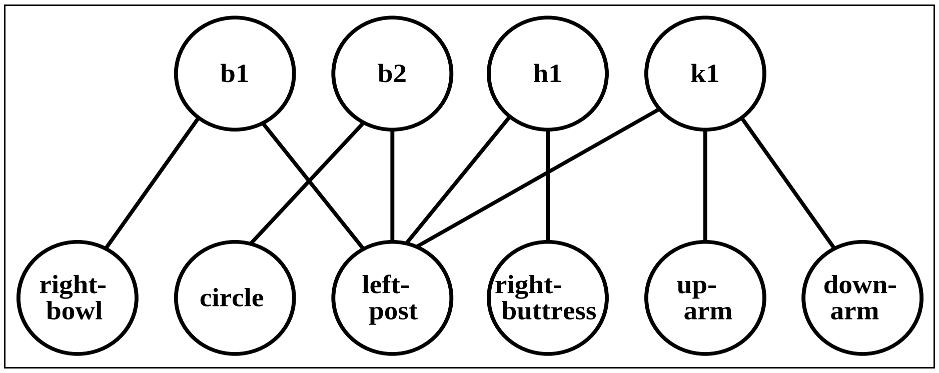

Figure 5.1: Part of the Conceptual Network.

The Conceptual Network is less sophisticated than the Slipnet seen in other FARG architectures [Mitchell 1990; French 1992; Marshall 1999], in that it relates concept nodes in a fixed way, whereas the various Slipnet implementations allow the relationship between concepts to shift dynamically throughout a run. Replacing the Conceptual Network with a Slipnet might be a possible direction for future work on Letter Spirit.

Coderack

In each of the Letter Spirit modules, as in earlier FARG models, most processing involves the use of the Coderack, which may be the most distinctive aspect of the general approach common to all FARG programs. In a traditional deterministic algorithm, procedure calls are made according to the order in which the flow of execution moves through a program, sometimes in straight-line fashion, sometimes looping, sometimes moving up and down through a hierarchy of routines and subroutines. The execution corresponds so precisely to the source code that one often uses the term "program" without feeling the need to specify whether the referent is the source code or the activity that a computer running the corresponding code undergoes.

In the FARG architecture, the closest counterpart to a procedure call is a codelet, which is a relatively short routine that performs some small operation, no one of which does very much of the program's work. This is only a computer-science perspective on codelets — they are also intended to correspond meaningfully to small-scale cognitive events, although it has not been proven that they correspond exactly to elements of actual human thought.

The Coderack is the repository of codelets, and execution begins with the nondeterministic selection of one codelet from the Coderack. This codelet is removed from the Coderack and then executed. This activity is repeated until either the Coderack is empty or some other condition for halting is reached.

A run of any of the Letter Spirit modules begins with the placing of some codelets on the Coderack. As codelets are run, the number of codelets remaining on the coderack generally decreases, but codelets may, as part of their work, add more codelets to the Coderack. The length and composition (in terms of what types of codelets are present) of the Coderack both change over the course of a run. How this occurs varies widely among the modules and can depend upon the events that play out during a run.

The process of selecting a codelet is performed nondeterministically, with each codelet having a certain chance of being selected. A given codelet's chance of being selected is a function both of the urgency it is given when posted and of the level of the temperature (explained shortly).

An Examiner run begins with the segmentation of the gridletter into parts (the details are provided later). The Coderack is then initialized with one gestalt codelet, which looks for hints about the possible letter category of the gridletter, and, for each part in the segmentation, two looker codelets, which begin the process of identifying what roles the parts may be fillers of. After that point, codelets themselves place new codelets on the Coderack, as previously-posted codelets are removed and run. The Coderack operates a bit like a secret society that is always replenishing itself, with new members added by the old ones that are leaving.

There is a simple top-level loop that with each pass through the loop selects a codelet from the Coderack, removes it, and then runs it; there are also a few extra-coderack actions that are occasionally carried out.

- If the activation of a role-set is over 99.0, the program quits, returning the highly active role-set as the winner (if more than one role-set is that active, one of them is picked at random).

- Every 800 codelets marks the beginning of a new phase, at which time the definitions of roles are "loosened" somewhat, allowing progressively more diverse parts to be recognized as fillers of roles. (More details regarding phases are offered later.)

- If the Coderack is empty, either more looker codelets are posted to it (if parts do not have the maximum number of labels) or the input is resegmented, to try something new.
- If 8000 codelets have run, the Examiner halts without offering any answer.
- A run can also end if the Examiner's measure of "goodness" reaches a certain value. This numerical measure, which is essentially the inverse of the temperature, is described below.

Temperature

The temperature is a number between 0 and 100. This is an inverse "goodness" rating for the quality of the work done thus far in a run, with high temperature corresponding to situations where the system has not yet built up much useful structure. It is a function of how far the program has come in returning an answer, and consequently it is calculated by determining the extent to which one role-set's activation in the Conceptual Network dominates that of all other role-sets.

Temperature is used by the top-level loop that runs the Coderack in order to determine the extent to which the urgency weights that are attached to the codelets should influence which one is selected. When the virtual roulette wheel is spun that picks the next codelet to be run, a high temperature causes the weights to be relatively disregarded, whereas at a low temperature, the weights are quite influential. As a result, the Examiner is more directed when an answer seems close, and is more likely to consider a wide range of options when it seems that it is far from reaching an answer, or is on the wrong track.

5.2.3 Codelets

Codelets carry out almost all of the work done during a Letter Spirit run. The nature of the Examiner, like that of any FARG model, depends enormously upon the specifics of its set of codelet types, listed below. Any codelet on the Coderack during an Examiner run is an instance of one of these. For many codelet types, instances are placed on the Coderack with certain parameters (such as a particular part) already specified. Other codelet types take no parameters. What follows is a description of each Examiner codelet type in a pseudocode fashion. The name of the codelet type is followed by the arguments (if any) that are passed to it, and a brief account of what function the codelet has.

gestalt codelet (no arguments)

A gestalt codelet is posted every time the input is resegmented. When a gestalt codelet runs, it adds a small amount of activation to each role-set node in the Conceptual Network, with the aim of bestowing positive activation to role-sets that seem to match the input (more to those that are more likely to be the correct answer), and negative activation to those that apparently could not serve as the correct answer. The gestalt codelet doles out activation to role-sets based on how well the input matches each letter category in terms of a few qualities, namely the degree of overlap with prototypes, the location of tips, whether or not the gridletter is a closed figure, and the presence or nonpresence of descenders and ascenders. Thus, the gestalt codelet is a crude letter recognizer in its own right, identifying likely answers based upon holistic properties of the input, rather than by following the decompositional approach of the Examiner as a whole. The activation that it adds to the Conceptual Network has the value of a heuristic, guiding other processing in the Examiner beneficially.

looker codelet(no arguments)

This codelet picks a part at random, and decides whether or not it is worth labeling. If the part has more than 2 tips or cannot, for other reasons, be seen as either a loop or a continuous path connecting two points, then a top-breaker codelet (see below) with high urgency is posted. If the part has any other problems (any of a few eccentric properties can, probabilistically, trigger this verdict), then it is marked as "whiny" and a pacifier codelet (see below) is posted. Otherwise (there is nothing wrong with the part), ten labeler codelets (again, see below) are posted for the part.

pacifier codelet (no arguments)

If there are no parts marked "whiny", then this codelet does nothing. Otherwise, it selects one of the whiny parts at random and handles it in one of the following ways:

- If the whiny part has more than 2 tips or cannot, for other reasons, be seen as either a loop or a continuous path connecting two points, then a top-breaker codelet (see below) with high urgency is posted.

- If the whiny part is not in direct contact with any of the other parts into which the input has been segmented, then the mark of "whiny" is removed from the part, and ten labeler codelets are posted for it.

- If the whiny part is not too small (a judgement which is made probabilistically), then the mark of "whiny" is removed from the part, and ten labeler codelets are posted for it.

- If a bigger part could be created by fusing the whiny part together with one of the neighboring parts (one that is in direct contact with it), without the new part having three or more tips, then the whiny part and its neighbor are stuck together, all the activations in the Conceptual Network are dampened to very near zero (each, however, retaining its sign, if nonzero), a gestalt codelet is posted, and two looker codelets are posted for the new part.

- Otherwise, the mark of "whiny" is removed from the whiny part, and ten labeler codelets are posted for it.

labeler codelet (argument: a part)

One of the dimensions for labels (e.g., height, width) is chosen at random and the appropriate label that describes the part in terms of that dimension is attached to the part. This step is iterated a total of nine times for each labeler codelet, and can thus attach a total of up to nine labels (fewer are attached if the same dimension is selected more than once, or if a dimension that is chosen had already been handled by an earlier labeler codelet). When all of this is done, a label-checker codelet is posted for the part.

label-checker codelet (no arguments)

A part is selected at random. If the part has enough labels (how many depends upon which phase the Examiner is in; in the first phase, the maximum possible number of labels and thereafter, one less than that), then a sparker codelet is posted for the part. Otherwise, nothing happens.

top-glommer codelet (argument: a part)

If a bigger part could be created by fusing the whiny part together with one of the neighboring parts (one that is in direct contact with it), without the new part having three or more tips, then the whiny part and its neighbor are stuck together, all the activations in the Conceptual Network are dampened to very near zero (each, however, retaining its sign, if nonzero), a gestalt codelet is posted, and two looker codelets are posted for the new part.

top-breaker codelet (argument: a part)

If the part still exists (it may have been glommed or broken since the codelet was posted), then the part is broken into two. All the activations in the Conceptual Network are dampened to very near zero (each, however, retaining its sign, if nonzero), a gestalt codelet is posted, and two looker codelets are posted for the new part.

sparker codelet (argument: a part)

If the part still exists (it may have been glommed or broken since the codelet was posted), then the part is handled in one of the following

- If the part serves fairly well as a filler for any roles (based upon how well the labels attached to the part match roles' norms), then those roles are sparked — namely, they receive activation and are bound to the part. Roles that are already active but not bound to any other part are more likely to be sparked. An activation-spreader codelet is posted if at least one role is sparked.
- If the part does not have enough labels (how many depends upon phase), then three labeler codelets are posted for the part.
- Otherwise, one of the following three actions promoting resegmentation of the input is chosen at random: either a top-breaker codelet is posted, a top-glommer codelet is posted, or the gridletter is resegmented based on gestalt.

activation-spreader codelet (no arguments)

If every part has at least one role bound to it, then there is a one-third probability that all the activations in the Conceptual Network are dampened to very near zero (each, however, retaining its sign, if nonzero), a gestalt codelet is posted, and one of the following three actions promoting resegmentation of the input is chosen at random: either a top-breaker codelet is posted, a top-glommer codelet is posted, or the gridletter is resegmented based on gestalt.

Whether or not any of the above measures are taken, activation is spread along the links in the Conceptual Network, and any role-sets whose activation exceeds a certain threshold is subjected to r-role tests.

5.2.4 Parallel terraced scan

One of the key ideas underlying earlier FARG models is that of the parallel terraced scan. It is also implemented as a feature in each part of the Letter Spirit program. The concept can be explained as follows.

One of the insights underlying integral calculus is that the area under a curve can be approximated by fitting rectangles onto the Cartesian plane, between the curve and the x-axis, and calculating the sum of their areas. The thinner the rectangles, the better the approximation to the actual area bounded by the curve. In a Coderack architecture, likewise, the small grain size of processing by codelets offers the potential for a tight fit to the mental processes that the architecture models.

Individual codelets are intended to model individual mental events on the sub-cognitive level, as has also been pointed out with respect to the Metacat extension to Copycat [Marshall 1999]; in the terms of Chapter 2, this means that they are meant to correspond, if only approximately, to the activity of a distributed mechanism. However, the operation of a FARG model may also be viewed on a higher level, in which some codelets may be seen as allies working towards a common goal, pitted against other groups of codelets implicitly working towards other goals. In the Examiner, this higher level of description may be considered a model of the cognitive level (in Marshall's terminology), which is conscious and corresponds to Chapter 2's algorithmic mode.

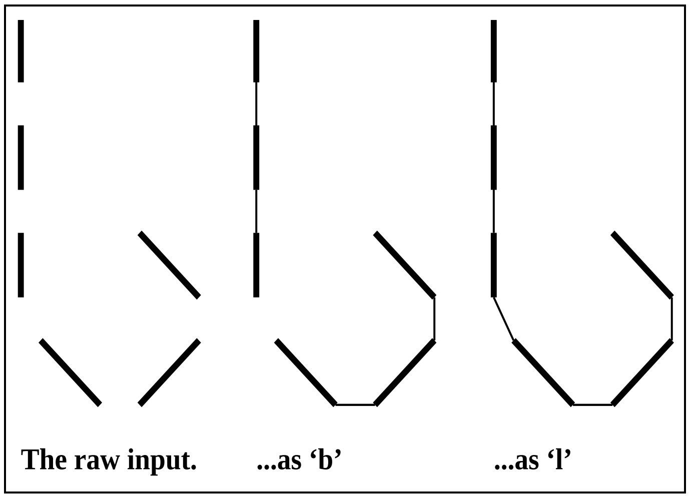

Figure 5.2: This gridletter can be segmented either as ‘b’ or as ‘l’.

To illustrate this, consider an example in which the Hint Four ‘b’ is given to the Examiner (as seen in Figure 5.2). This is an ambiguous letterform that may be recognized as either a ‘b’ or (despite the designer’s intent!) an ‘l’. Early in the run, a gestalt codelet will run, and ‘b’ and ‘l’ will each receive considerable positive activation from it. Almost every other letter category will have its activation set to a negative value, essentially ruling it out as the final answer. The initial segmentation of the gridletter will lead to an attempt to parse it as one of the two letter categories with active role-sets, and both are possible in different runs. For our current example, let us say that it is segmented as a ‘b’ (as in the middle). At this point, the Coderack contains many labeler codelets, which will label both parts. If both parts are labeled sufficiently, and if sparkers then fire for both parts, the roles for ‘left-post’ and ‘right-bowl’ will likely be activated and bound to the parts. Then activation of the role-set ‘b1’ and identification of the letter as a ‘b’ are liable to take place very quickly thereafter.

While any labeler codelet, individually, is agnostic with regard to letter category, each of these codelets, the savvy onlooker can say, is promoting 'left-post' and 'right-bowl' as candidates for the roles underlying the two parts, and thus 'b' as the final answer. In the same time frame, sparker codelets can run, and, until the parts are adequately labeled for sparking, the Examiner may choose to resegment the gridletter, which may lead to it being left as one large part, which in turn favors a final answer of 'l'. If such a resegmentation does occur, all labelers posted to the Coderack at that point will attempt to label the one part in ways befitting the central post role of 'l'.

In no sense do any individual codelets ever explicitly favor any role-set, but given the context of a particular segmentation of a gridletter, the execution of a codelet can be seen to implicitly further the cause of a particular role set. In other cases, the situation may be ambiguous; it may be harder for observers and run-time analysts of the Examiner to determine which of various outcomes an individual codelet favors. The point is that higher-level activity does occur in the Examiner, and it is driven by the low-level activity of many codelets. This is a prime example of what is often called emergent behavior.

A goal of the FARG architecture is to create models that search many possibilities at once, but with preferential effort directed towards answers that are more promising. This kind of search is called a parallel terraced scan [Hofstadter and FARG 1995]. In the case of the Examiner, it is through all the roles, role-sets, and letter-categories that search takes place. However, the activity on the lower level — that of codelets — is what makes the parallel terraced scan work. In the example above, with the gridletter segmented as on the left, categories 'b' and 'l' may each have codelets acting as their advocates on the Coderack, but 'b' has an advantage over 'l', and from that point on, it will more likely end up as the final answer. In principle, more outlandish answers may be possible, although they have far lower probabilities of being selected as the final answer. Other gridletters may balance three or more possibilities fairly evenly, though usually a gridletter will have a small number of letter categories as realistic possible answers.

This is not the only manner in which the architecture enables the parallel terraced scan to operate. In fact, as will be discussed at greater length later, most of the optimizations made to the Examiner after 1995 were inspired by the goal of making the parallel terraced scan operate in as many ways as possible.

Temperature: A general phenomenon

The use of temperature in FARG models as a means of directing breadth of randomness in search resembles similar mechanisms in other approaches to cognitive modeling, including simulated annealing and Boltzmann machines [Hinton and Sejnowski, 1986]. The generality of the approach cannot be overstated. Individuals, communities of individuals, and probably even lower animals are apt to widen their search when the chance of success seems to be diminishing.

A large-scale example is provided by the voting patterns of an electorate. Figure 5.3 shows the pattern of voting in Germany's Reichstag elections in years before and after the onset of the Great Depression, which hit Germany very hard at that time [Shirer 1960]. Votes cast for the parties on both extremes of the political spectrum rose at the expense of the seven or so parties comprising the middle. The share of the vote granted to the extremes nearly quadrupled from 1928 to 1932. The general principle is that in circumstances where measures that are initially preferred fail, options of lower initial interest start to receive higher consideration, and the steep preference for the center (as seen in 1928 in the figure) flattens out. This is exactly the effect that high temperature is intended to have in the selection of codelets from the Coderack, and, therefore, from the high-level perspective, in the consideration of possible answers.

Temperature is not as important to the Examiner as it was to the early FARG models [McGraw 1995, p. 182], and is virtually a non-factor in the other Letter Spirit modules. The parallel terraced scan does occur in those modules, and in Letter Spirit as a whole (via a mechanism similar to temperature), but by different means.

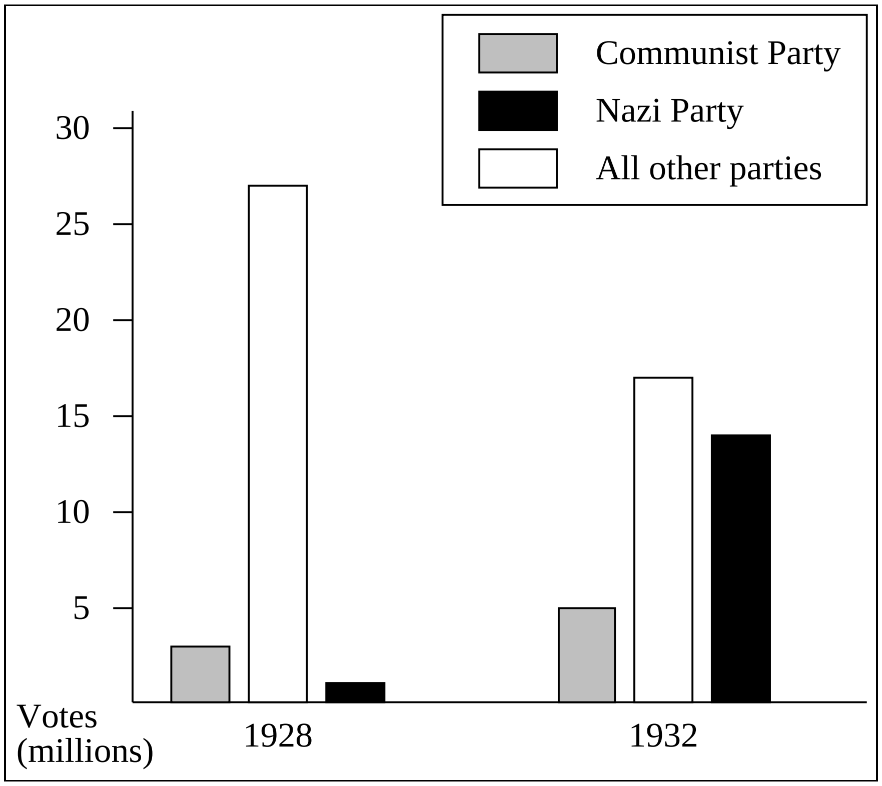

Figure 5.3: Germany, 1932. Hard times lead to exploration of the extremes, deemphasizing the middle. This is a demonstration of the effects of high temperature upon decision-making.

5.3 Performance

The Examiner has been tested on a variety of sets of gridletters for the purpose of evaluating its performance. For consistency within this thesis, the set of 23 sample gridfonts introduced in Chapter 1 was used as the test set. For comparisons with past work, namely that in [McGraw 1995], a different but overlapping test set was also used.

Accuracy (the proportion of correct answers) is the most important measure of performance. In fact, the accuracy of the Examiner is probably the single most important factor in determining the quality of Letter Spirit output, because mistakes in categorization undermine the work of every other module.

Speed is also an important pragmatic consideration. When the Examiner is used as a stand-alone program, with single runs demonstrating its workings in a graphical display, it makes little difference if a run takes a second or a minute. In fact, runs that are too fast are difficult to watch. However, a full Letter Spirit run can involve over 300 runs of the Examiner. A one-minute difference per Examiner run would thus extend a Letter Spirit run by over five hours.

A third measure of Examiner performance is how well it models human performance, making the same kind of errors that human subjects make. The third measure, insofar as it diverges from accuracy, was considered subordinate in importance to the first two, and the post-McGraw optimizations to the Examiner decreased, to some extent, the correlations between Examiner output and data from human subjects.

5.3.1 Sample runs

Figure 5.4 shows a sample run involving the Examiner. This was a short run on a letter that is easy to recognize. On the left is the input gridletter (in these illustrations, spaces that are not part of the grid have been inserted between quanta, to make room for the display of ligatures that indicate how quanta are grouped into parts). The first two codelets activate role-sets ("wholes") based on the letter's gestalt, and then segment the gridletter under top-down influence of the most active letter category, which is 'b'. Over the next 24 to 29 codelets, the two resulting parts are labeled, and then sparked. The next time that activation is spread (which happens to be four codelets later), the correct role-set receives high positive activation from its constituent roles, and recognition is complete. This is an ideal Examiner run, in which the initial segmentation and subsequent labeling lead to prompt sparking of roles that support correct recognition. At the end of the run, the activation of the role-set 'b2' is the maximum of 100.0.

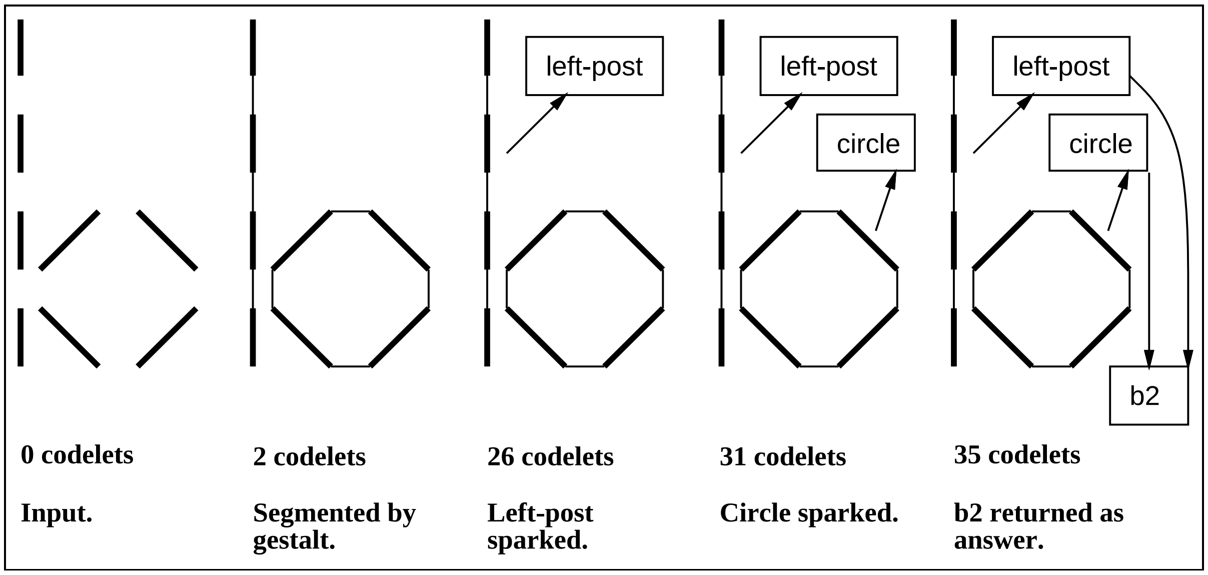

Figure 5.4: The Examiner recognizes a 'b'.

Figure 5.5 shows a gridletter that the Examiner has more trouble with, and one that people undoubtedly find harder to recognize as well. The sample run on this gridletter was approximately 100 times the length of the last example, and cannot be summarized succinctly in a diagram. The following discussion will step through the run.

Here, the gestalt codelet gives more activation to 'o' role-sets than to 'a', by the small margin of 15.8 to 14.7. The initial segmentation, which suggests 'o', places all the quanta together in one part, shown in the figure as Segmentation 1. Parts with exactly one tip, however, have a significant chance of triggering resegmentation, and this happens very quickly, leading to Segmentation 2. The new segmentation, however, also has a one-tip part, and by codelet number 8, the program has already moved on to Segmentation 3. The lower loop is labeled and sparked as a down-circle, but the remaining upper parts do not spark any roles, and each attempt at sparking

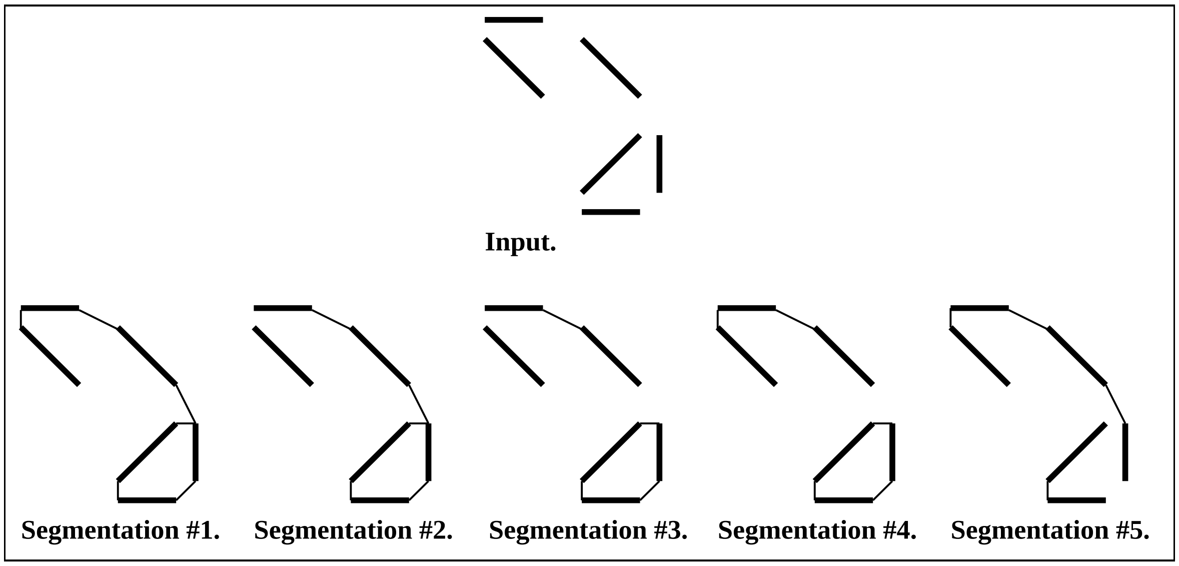

Figure 5.5: The Examiner has more trouble recognizing this ‘a’.

brings a chance of resegmentation. This finally occurs by codelet number 69, which goes back to Segmentation 1. For the same reasons as before, this does not work out, and a series of breakings, glommings, and gestalt-driven resegmentations leads to Segmentation 4 after codelet number 97.

This is the segmentation that eventually led to recognition as an ‘a2’ (consisting of a closed ‘down-circle’ role and an ‘a-arch’ hanging over it), but in this early stage of the program, this cannot take place. The lower loop is again sparked as a down-circle. However, the upper part is not enough like the Conceptual Memory’s description of an a-arch to spark that role. By codelet 130, the gridletter is resegmented. Segmentation 5 promises the opportunity to recognize the gridletter as an ‘a1’ (again, with an ‘a-arch’ above and to the right, but with an open ‘left-downbowl’ beneath that), rather than an ‘a2’. This manner of segmentation is less likely than the previous ones, but it did occur by codelet 615, after many other segmentations had been attempted without recognition. Segmentation 5, however, happens to suffer the same problem as Segmentation 4 — that the resulting parts are too unusual to spark the correct roles at this point in the run. A human observer rooting for the Examiner to successfully categorize this gridletter as an 'a' may groan at this setback, and can only wait for the program to get back to that point. However, whether it does so or not does not actually matter in the short term, because with the initial threshold for sparking, the top "limb" of the gridletter cannot be classified as an a-arch.

Every execution of 800 codelets is considered a new phase in the Examiner, and at such points the threshold for sparking is lowered by 10% of the previous value. Therefore, throughout a run, the sparking threshold decays exponentially.[1] With a threshold of zero, any part may be classified as any role. However, this never happens; with a maximum of 8000 codelets per run, even by the end of a run, the threshold gets down to no less than 38% of its initial value. Another effect of the 800-codelet phases is that the number of labels that a part must have attached to it for sparking to take place is lowered from the maximum possible number of labels to a number lower than that by one (this is not continuously decremented with each phase, but is lowered only once). The changes that occur at the onset of these phases constitute the "loosening" referred to earlier in this chapter. Because loosening occurs only at the beginning of discrete phases, a knowledgeable observer can already tell that for this example, the Examiner has essentially no chance of recognizing the gridletter until after codelet 800.

In vain, the Examiner resegmented the gridletter a total of 23 times before codelet 800. Each time, the resulting parts failed to spark the roles making up any role-set. The observer might suspect that when the sparking threshold is lowered, new sparkings will be possible, and that recognition might succeed. In this case, however, the second phase did not lead to enough loosening for that to take place. Segmentation

[1]Actually, in later phases, the exact value used alternates randomly between higher and lower values. The later section on the optimizations made to the Examiner after 1995 explains exactly how the threshold is used.

4 led to successful sparking of the down-circle, but not the a-arch. Segmentation 5 led to successful sparking of the a-arch, but not the left-downbowl. 18 resegmentations took place throughout the second phase of the run, none leading to any more success than the resegmentations during the first phase.

The same story was repeated many times throughout the third phase of the run, and also through the fourth. When the fifth phase began at codelet 3200, the threshold had finally been lowered enough for recognition. Early in the fifth phase, the program went through a couple of unusual segmentations (not shown). At codelet 3286, Segmentation 1 came up, and when that failed, Segmentation 4 arose at codelet 3312. The lower portion was sparked as a down-circle soon thereafter, and by codelet 3386, the upper portion was sparked as an a-arch. After this lengthy prelude, recognition of the gridletter as an 'a2' finally took place four codelets later, at which time the 'a2' node in the Conceptual Network had an activation of 53.0.

Why phases?

We are all agreed that your theory is crazy. The question which divides us is whether it is crazy enough to have a chance of being correct. My own feeling is that it is not crazy enough.

— Neils Bohr, to Wolfgang Pauli

In this run, and in many like it, the length of the run was greatly prolonged by the sparking threshold being too high, and the intervals at which lowering takes place being too long. However, for many other runs, a lower initial sparking threshold can cause problems. Given a low enough sparking threshold, any part can spark any role, and therefore almost any gridletter could be recognized as a member of almost any letter category. With low initial thresholds, an increased number of false identifications occur fairly quickly. Making the length of the phases too short causes a related problem for some gridletters. In these cases, finding the correct answer depends upon a segmentation that comes up rarely. If the phase is too short, it will be apt to end before the correct segmentation is tried; when the program moves on to the next phase, the sparking threshold is lowered, which can lead to the same problem as with a threshold which is initially too low. The three parameters relevant to phases (the initial sparking threshold, the length of each phase, and the amount by which to decrease the threshold when a new phase starts) were carefully set with values that appear to maximize overall performance.

5.3.2 The 23 example gridfonts

Walkthroughs for two sample runs show, in detail, how the Examiner successfully recognizes gridletters. A different way of looking at Examiner performance is to look at tests performed on a large collection of gridletters. This reveals which gridletters the Examiner *cannot* recognize, and offers a complementary look at how the Examiner works, which in turn reveals how good the Examiner is as a model of human cognition.

Raw performance

An extensive test was conducted wherein the Examiner ran ten times on each gridletter of the example gridfonts (almost six thousand runs in all). Overall performance was respectable, at 79.0% accuracy. Table 5.2 shows, for each gridfont, how many runs resulted in correct recognition, how many in incorrect recognition, and how many quit without returning an answer. There is a great deal of variety across gridfonts, with ten of the gridfonts recognizable at 90% or better. Performance varies over a wide range from gridfont to gridfont, as it would for any program or person tested on a similarly varied set.

Difficult gridfonts were included in the test set specifically to probe shortcomings of the Examiner. Performance on Sluice and Three-D is particularly poor, as these gridfonts involve abstractions that are not built into the Examiner. Three-D could be recognized by a program only if it understood three-dimensionality, and Sluice only by a program that could perceive non-continuous role-fillers. Because of the poor showing the Examiner makes on these two gridfonts, they cannot be part of tests of the other parts of the program; every part of Letter Spirit depends upon decent performance on the part of the Examiner. It is shown below that the overall performance of the Examiner is nonetheless quite strong, even in comparison to that of people.

	Right	*Wrong*	*Quit*
Benzene Left	95.4	4.6	0
Benzene Right	99.6	0.4	0
Boat	97.3	2.7	0
Bowtie	60.4	38.1	1.5
Checkmark	76.5	19.6	3.8
Close	94.6	5.4	0
Double Backslash	83.8	14.6	1.5
Flournoy Ranch	76.5	23.5	0
Funtnip	54.2	32.7	13.1
Hint Four	88.8	11.2	0
House	94.2	5.8	0
Hunt Four	95.8	4.2	0
Intersect	63.5	36.5	0
Sabretooth	81.9	18.1	0
Shorts	97.7	2.3	0
Slant	90.8	9.2	0
Slash	85.8	13.8	1.5
Sluice	13.5	86.5	0
Snout	96.2	3.8	0
Square Curl	81.9	18.1	0
Standard Square	100.0	0	0
Three-D	0	0.8	99.2
Weird Arrow	87.7	12.3	0
Total	79.0	15.8	5.2

Table 5.2: Examiner performance on 23 gridfonts.

Comparative performance

Table 5.3 shows the performance of the current Examiner, as well as that of other programs, and that of the subjects who participated in a gridletter-recognizing experiment in the summer of 1993. The set "ALL" contains 545 gridletters, and it overlaps somewhat with the 23 example gridfonts used thus far. Examiner '95 is the Examiner as reported in [McGraw 1995]. Netrec+ is a simple connectionist model — a three-layer backpropagation network trained on a large number of gridletters and then tested on one subset of ALL. Gestalt is the function used in the Examiner's gestalt codelet, used here as a standalone recognition program.

	NORMALS	*ALL*	*TEST*	*Codelets (ALL)*
Examiner '99	98.5%	93.5%	91.6%	408
Examiner '95	95.5%	82.4%	—	1389
Netrec+	—	—	79.2%	—
Gestalt	—	—	65.6%	—
Humans	90.1%	84.0%	80.1%	—

Table 5.3: Examiner performance versus other programs, and people.

The first column shows performance, in terms of percent correct answers, on an easy subset of the data. The second column shows percent correct on ALL. The third column shows percent correct on a test set that is ALL minus some of its very easiest gridletters, which were used for training some programs that learned. The exclusion of the training set made TEST the desired set for across-the-board comparisons. Finally, the fourth column shows the mean number of codelets in a run. There are a number of blanks in the table, as not all programs were tested on all data sets.[2]

[2]And also, how to count the number of codelets a person has run is still an unsolved problem.

The clear single result is that the current Examiner is by far the dominant gridletter recognizer among this field of five. It never does worse than make half the number of errors that its closest competitor does, on any of the subsets tested. This shows the strengths of the approach taken here. The improvement over the 1995 Examiner is considerable, and the reasons for it are discussed in this chapter's conclusion. The comparison to people is rather interesting. First, it should be noted that these people are not necessarily experts, as people go, in reading typefaces that strain legibility. Second, the improvements made since 1995 move the emphasis towards the goal of simulating expert letter-readers, as opposed to randomly-selected subject-pool individuals such as those tested in the experiments described in [McGraw 1995] and in Chapter 2.

The performance of the gestalt function utilized within the Examiner is worth noting. It is the worst of the group; however, it does its job adequately provided that it selects the correct answer as *one of* its top answers. Whether or not it chooses the correct answer as its *top* candidate does, though, affect Examiner speed. Part of the considerable speedup, in terms of codelets, that came as a result of the optimization is owed to the fact that the gestalt function is usually correct, which allows the Examiner to segment the gridletter correctly in the first couple of codelets of most runs.

5.3.3 Errors by the Examiner

All happy families are the same; all unhappy families are unhappy in their own way.
— Leo Tolstoy, *Anna Karenina*

Like families, Examiner runs can go right in only one way (with prompt correct recognition of the input)[3] but can go wrong in many ways. The Examiner's performance on the 23 example gridfonts can be seen on a gridletter-by-gridletter basis in Figure 5.6, Figure 5.7, and Figure 5.8, which appear at the end of this chapter. Each gridletter was tested ten times. Gridletters in dark boxes were classified correctly four or fewer times out of ten; the ones in dark boxes with heavy outlines were not classified correctly even once in all ten tries. Gridletters in white boxes were classified correctly five to seven times. The remaining gridletters were classified correctly eight or more times.

A set of gridletters that leads to a high number of errors provides a good survey of the ways that things can go wrong for the Examiner. Table 5.4 lists some cases that are representative of nearly every kind of gridletter that gives the Examiner trouble. The same gridletters are presented graphically in Figure 5.9. Many of these gridletters are jarring to the eye, and the fact that they might cause a letter recognizer, whether computer or human, some difficulty is probably not surprising. With the details of Examiner runs available, the reasons why the Examiner stumbles on these examples can be provided. It is worthwhile to consider some of these errors, letter by letter.

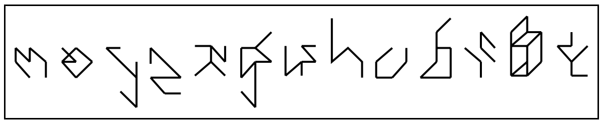

Figure 5.9: Gridletters that the Examiner does not often recognize correctly.

The Benzene Left 'm' is usually seen as an 'n' by the Examiner (it was also recognized correctly two times out of ten) for a number of reasons. One is that it really is less of a lowercase 'm' than an uppercase 'M'. As a consequence, the gestalt function prefers it as an 'n'. Therefore, the initial segmentation tends to be as 'n'

[3]Universal statements beg for exceptions, and this one is no exception. It will be explained shortly that some runs produce the right answer, but still leave something to be desired.

(into two parts), and it can receive the proper segmentation only if recognition as an ‘n’ fails. However, it makes a pretty good ‘n’. If this gridletter is broken into two parts, with the break falling at the dip in the middle, the left piece is a good left-halfpost, and the right piece a good right-buttress. Frequently, this segmentation leads to recognition as an ‘n’ before the more complex segmentation that can lead to ‘m’ arises through the relatively random process of breaking and glomming.

This example raises a more general phenomenon that arises in the Examiner. Often, a gridletter can hypothetically be segmented in more than one way, and therefore, sparking and the spread of activation can lead, in different runs, to multiple answers. One might expect a letter-recognition program, at least in its internal state, to somehow indicate the viability of answers other than the one it finally settles on. The Examiner’s gestalt function does this by returning a value for every possible letter category, along with its assessment of how well the input serves as a member of that category. The Examiner as a whole, however, may be fully capable of recognizing a gridletter as a member of a certain letter category in one run, and yet in another run on the same gridletter fail to activate that category, either in the Conceptual Network, or in any other way. This is because all aspects of recognition in the Examiner, other than via gestalt, depend upon segmentation. Only one segmentation is considered at a time, and as soon as a segmentation leads to correct recognition, the run ends, leaving any other possible answers uninvestigated. Shoring up this “tunnel vision” on the part of the Examiner would be an excellent direction for future work.

Bowtie ‘o’ is a bit unusual as an ‘o’, to be sure. The Examiner calls it an ‘a’ every time, because its gestalt (especially given the position of its one tip) suggests ‘a’ first, and then the spur off to the left (despite its quirky jaunt in, then out of the closed area) is easily sparked as an “a-arch” while the diamond sparks the “circle” role, and those two sparkings are enough to lead to recognition. This highlights the fact that the Examiner must account for any spare material hanging around the letterform. One of the r-role checks, as was noted earlier, makes sure that the answer's role-set's role-fillers account for all the material in the gridletter. A person (such as the designer) can easily see this gridletter as an 'o' with the spur as an add-on feature (spurious, one might say). The fact that the Examiner does not allow such perception is a current limitation, but it is one that prevents many errors in the opposite direction, if an important part of the gridletter could be ignored (for example, most 't's could be seen as 'l's if one or two quanta were ignored).

The next example, Checkmark 'g', is a good example of another class of error — namely, those errors that are hard to call errors! This gridletter was not part of the experiments probing human gridletter recognition, but it seems safe to say that many people would agree with the Examiner in calling this a 'y'. In the context of Checkmark, however, this is not an outrageous choice on the part of the designer, because Checkmark emphasizes adherence to the style over letter-category strength, and is not intended to be highly legible.

Double Backslash 'z' is not intrinsically such a bad 'z', but it violates the expectation that a 'z' should be fully contained in the central zone. The Examiner *can* see this gridletter as a 'z', but also commonly gives answers of 'y' and even 'n'. The correct answer and the two incorrect answers alike require the Examiner to stretch something. For this to be recognized as a 'z', the large intrusion into the descender zone must be overlooked. For 'y', three allowances (what FARG terminology frequently calls "slippages") must be made: the weakness of the portion in the central zone as what is call a "left-uparc", the fact that the portion in the descender zone curves more or less the opposite way from what is expected of a 'y's tail, and the fact that the tail does not extend above the point where it meets the left-uparc. While the 'y' answer requires *more* slippages than does the 'z' answer, this does not have a bearing on how the Examiner answers. The Examiner loosens roles according to phases, and if the three 'y' slippages are all permitted when (or before) the phase when the one ‘z’ slippage is permitted, then both ‘y’ and ‘z’ become viable answers at the same time. Or, if all three ‘y’ slippages are permitted before the one ‘z’ slippage, then ‘y’ may be a more frequent answer. This is another manifestation of tunnel vision — the Examiner returns the answer it gets to first, not the one it likes best.

Flournoy Ranch ‘x’, weird in various ways, has one characteristic that makes it completely impossible for the Examiner to recognize it as an ‘x’. That is, the only way to parse it as an ‘x’ role-set requires that one quantum be shared by the two role-fillers. This is simply impossible in the Examiner at present. Segmentation in the Examiner divides a letterform’s quanta into distinct, nonoverlapping parts. This unfortunately makes the recognition of certain gridletters impossible.

Funtnip ‘g’ is within the Examiner’s ability to recognize, when it gets lucky enough to segment it correctly. This happened one time out of ten tests, and then, only very late in the run. There are simply a lot of ways to segment this gridletter, and only one way that allows recognition as a ‘g’. This segmentation is not produced very often, and recognition requires a lucky series of breakings and glommings to come up with the right one.

Funtnip ‘u’ is another victim of the Examiner’s tunnel vision. Based on the locations and directions of the tips, the gestalt function favors ‘r’ as the answer. Odd as the slippages required to make this gridletter acceptable to the Examiner as an ‘r’ may be, they are all allowed in the first phase. The Examiner sees this as an ‘r’, very quickly, ten runs out of ten.

Hint Four ‘h’ demonstrates one way that low-level vision in the Examiner differs in a fundamental way from human low-level vision. The sharp $45°$ angle pointing to the lower left suggests a tip to the human eye; this effect was discussed in Chapter 1. The Examiner sees continuous paths as continuous paths, with no tips in their midst, regardless of sharp turns and corners. Consequently, the Examiner sees this as an ‘l’, though a slightly odd one, and it almost always returns that answer very quickly.

House ‘v’ is inherently ambiguous, floating between ‘u’ and ‘v’. The definitions of the roles relevant to those two letters leads the Examiner to virtually never spark “left-wing” or “right-wing”, the ‘v’ roles, with parts that have a horizontal quantum on the baseline. For that reason, House ‘v’ is always categorized easily and quickly as a ‘u’.

Slant ‘b’ takes tunnel vision to an extreme, because this gridletter suggests nothing but ‘b’ to the human eye. The gestalt function, however, is misled by the tip in the upper *right*, and hence it targets this gridletter as a ‘d’, and that choice determines the initial segmentation. Taking the path from the tip down to the lower right corner as the post and the rest of the gridletter as the bowl creates two rather quirky parts, but the Examiner allows this level of quirkiness even in the first phase, so recognition as a ‘d’ is prompt, virtually guaranteed, and rather perversely precludes the later finding of the far better answer, ‘b’.

Two possible changes to the program could help the Examiner identify Slant ‘b’ correctly. One would be if the initial segmentation of the gridletter were nondeterministic, and did not automatically elect the highest gestalt. Then, the fact that the gestalt function prefers this as a ‘d’ would still lead to the wrong answer a high percentage of the time, but not always. A second would be if roles were simply made tighter (that is, if the sparking threshold were higher) in the first phase. Both of these changes have merit, and both would benefit other gridletters besides this one. However, as with many design decisions, it amounts to a tradeoff, because the changes that would allow correct recognition of this gridletter would harm Examiner performance on others — not so much in terms of accuracy as speed. The emphasis on the speed of the Letter Spirit program sacrifices accuracy in a few cases, and Slant ‘b’ is one of those.

Sluice ‘f’ is just one of many gridletters in that font alone that the Examiner cannot recognize, due to the fact that seeing it as an ‘f’ involves two role-fillers that are non-continuous. This is similar to the problem in Flournoy Ranch, and dates back to the 1995 Examiner. McGraw [1995] suggested (p. 225) that giving the Examiner the ability to "hallucinate" imaginary quanta in the gaps might be one solution to this problem.

The Examiner did not once recognize any letter from Three-D correctly. The reason why is the inability of the program to produce representations of three-dimensional shapes based on two-dimensional input, and to rotate those internal representations to a perspective from which they appear to represent two-dimensional letters. Three-dimensional perception is a blatant example of a distinct cognitive ability without which correct categorization of certain kinds of stimuli is essentially impossible. Letter Spirit has some shortcomings due to some highly specific abilities such as this being absent from its implementation.

Weird Arrow 't' suffers from a problem almost the same as that of Hint Four 'h'. Again, a 45° angle (in this case, the "elbow" that juts out to the left) is meant to indicate something more than what the Examiner sees it as. Or, because the sharp angle is meant to indicate a point, one might say that it is meant to indicate something less. The Examiner is very rigid in its expectation that a role-filler should consist of a single-file path of quanta, end-to-end, contiguous, and not overlapping with any other role-fillers in the same letterform. A person might see the space *within* the sharp angle pointing to the left in Weird Arrow's 't' as its crossbar, essentially making a figure–ground reversal, with empty space between quanta standing for a role. The Examiner cannot do this. The Examiner does categorize this gridletter correctly a significant fraction of the time, but it is only because the spur to the right can be seen as a short crossbar (which does not actually cross anything), and the rest of the gridletter as the t-post, which is certainly not how people see it.

Table 5.4 summarizes these examples, and the wide variety of problems that the Examiner can encounter during a run that the errors demonstrate. It was stated

	Gridletter	*Answer*	*Reason*
Benzene Left	m	n1	easier to parse it as n1
Bowtie	o	a1	sees "serif" as a role-filler
Checkmark	g	y2	easier to parse it that way
Double Backslash	z	y1	reluctantly sees part as weird role-filler
Flournoy Ranch	x	k1	role-fillers cannot share a quantum
Funtnip	g	QUIT	cannot parse correctly
Funtnip	u	r1	three weird allowances not sufficiently disliked
Hint Four	h	l1	underestimates effect of collinearity
House	v	u1	flat bottom almost guarantees 'u'
Slant	b	d1	gestalt strongly suggests wrong answer
Sluice	f	QUIT	requires role-fillers to be continuous
Three-D	b	QUIT	an entire extra stage of processing is necessary
Weird Arrow	t	k1	figure–ground; the crossbar is traced, not drawn

Table 5.4: A sample of Examiner errors over the 23 example gridfonts.

earlier that Examiner runs can go wrong in many ways; these examples show the ways that account for the overwhelming majority of errors.

It should be added that there are two ways that an Examiner run can go wrong even in those runs that eventually produce the correct answer. The first way is that a run can simply take a long time. The earlier detailed examples of Examiner runs showed that one run (the one with an 'a') can take over 100 times as long (as measured in codelets) as another (the one with a 'b'). This wastes computational resources and a few such Examiner runs can significantly increase the runtime of an entire Letter Spirit run.

The second way of getting the right answer but not succeeding completely is illustrated by example in Figure 5.10. A single 'e' is parsed (segmented into parts, with the parts bound to roles) in two different ways that involve the same set of roles. However, the parsing on the right shows how an aberrant segmentation can nonetheless lead to a parsing that produces the correct answer. It is certainly odd to perceive this letter as having its crossbar on top of its e-bowl; the parsing on the left seems far more apt. However, just as the Examiner's tunnel vision can lead it to an utterly wrong answer, after which it halts without seeking the correct answer, an outcome such as that on the right is also possible, and it likewise precludes a subsequent search for the best answer. From the standpoint of letter categorization alone, this is not a problem, because the proper category has been identified in any case. The Adjudicator, however, depends upon receiving a good parsing from the Examiner; in the broader sense, the Examiner's quest is to come up with both the correct letter category *and* the correct parsing. In that sense, getting the right answer for the wrong reason is not getting the right answer at all. This sort of error is not very common, but it reveals an interesting complexity of the Examiner's task.

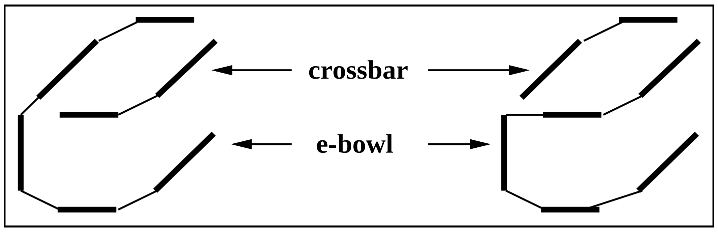

Figure 5.10: The right way and the wrong way to recognize an 'e'.

5.4 Conclusion

5.4.1 Categorization

The Examiner is a relatively complex model of categorization. Each run involves four distinct mechanisms for categorization. The initial gestalt codelet carries out categorization by means of a distributed process. Then, after segmentation and labeling, parts are categorized by sparker codelets, in a way that tightly corresponds to the *weighted feature bundle* model, discussed and rejected by Lakoff [1987]. Eventually, role-set and therefore letter category are selected based on conceptual activations that the program comes to by possibly complex machinations. This can be thought of as a third and fourth method of categorization, since runs requiring resegmentation employ considerably more complex operations than those that simply proceed directly from the initial segmentation.

An important point to be made here is that the kinds of categorization vary in nature. Lakoff [1987] cites the analyses of Eve Sweetser and of Eleanor Rosch, and offers analyses of his own (pp. 115-6), regarding why weighted feature bundles cannot account for categories such as those invoked by words like “lie” and “bachelor”. However, this only goes to show that *some* categories are not accounted for by weighted feature bundles. The account given so far in this thesis shows that categories can arise from different mechanisms, and thus that the possibility remains that some categories could be described by weighted feature bundles. Specifically, human preattentive categorization based on criteria of which the subject is not aware involves a distributed-mechanism sort of categorization, and this is perfectly consistent with the labeling–sparking categorization of parts as role-fillers that takes place in the Examiner. Lakoff’s rejection of this type of mechanism is too broad, as it is based on the properties of specific types of categorization that inherently involve conscious activity. As was seen with theories of letter recognition in Chapter 2, one must be careful in declaring the correctness of a complex theory that is to supplant an older one, particularly if the test data probe only one side of the distributed–algorithmic distinction.

The Examiner handles role-level categorization precisely as a weighted feature bundle, since the score calculated by the sparking codelet, which is compared to the sparking threshold, is the sum of weighted inputs. By contrast, the Examiner treats letter-level categories in a more complex manner. The strength of the Examiner’s performance suggests that weighted feature bundles may indeed be an adequate characterization of *role* categories even if a more sophisticated account is needed to explain and model *letter* categories.

5.4.2 Optimization

Meaningful activations

The revision of the Examiner from its 1995 version to the 1999 version described in this thesis had three main goals. The first was simply to make the program faster. A full run of Letter Spirit requires hundreds of runs of the Examiner, and a major speedup in the Examiner has meant a Letter Spirit program that, using the present hardware upon which it was implemented, designs a gridfont in hours (or less than an hour in some cases), as opposed to days. A second goal was to maintain, and if possible, enhance, its tight fit as a model of human performance. The same set of modifications that achieved one of these goals fulfilled the other admirably, and this work is described in detail by Rehling and Hofstadter [1997], but will be recapitulated here. Subsequently, as development of the Adjudicator proceeded, it became clear that the Conceptual Memory's representation of letters had to be retooled entirely for the Adjudicator to work properly.

Broadly speaking, the first two goals were accomplished by a number of changes to the program, virtually all of which served the primary goal of making any role or role-set's activation in the Conceptual Network, at all times throughout a run, reflect as closely as possible the evidence that that role or role-set is actually involved in the final answer. Because many activities in the Examiner depend upon activation levels, it is desirable to make changes in those levels as soon as possible, rather than to allow various subprocesses in the Examiner to be misled by activation levels based on outdated events. The information contained in the relative activations of roles is used to inform many processes in the Examiner.

The following six items summarize the changes made in the 1997 optimization of the Examiner:

- Whenever a gridletter is resegmented, all activations in the Conceptual Network are cleared (to near zero), so that leftover activations based upon sparking of parts that no longer exist do not muddle further processing.

- The gestalt codelet was made more accurate as a letter-recognition program in its own right, and was invoked at least once early in each Examiner run. This seeds the activations of wholes, and provides useful top-down pressure for all other work in the program.

- R-role checking, which used to be the task of a separate r-role checker codelet, was integrated into activation spreading. In essence, this replaced the posting of a codelet for *eventual* r-role checking (whenever the codelet happened to run) with immediate action.

- The onset of phases of loosening was made probabilistic. Before, distinct phases of loosening were enforced on a fixed schedule. With the modification, a short first phase maintains the tightest roles, and most letterforms are recognized during this phase. Subsequent phases randomly toggle between higher and lower looseness settings. Thus, a letterform that can be identified only with loose roles may be recognized relatively quickly, while a letterform that requires tight roles will have many chances later, if it is not recognized in the first phase.

- Originally, the activation of each role node and whole node was set to the previous activation plus the *sum* of the weighted inputs from each connected node, and then was reduced by a decay factor. Activation could only pass from a whole down to a role, however, if the activation of the whole was above +75. A large jolt of activation could stay with a node for a long time, and would decay only very slowly. This meant that the information contained in the activations would not accurately reflect the latest information that acts of sparking provide. In the optimization, at the time that activation spreads, a node is allowed to retain only a small portion of its previous activation, in addition to what activation is spread to it via its inputs. For a whole, the *sum* of the weighted inputs from all connected role nodes is added in. For a role, the *maximum* of the weighted inputs from the connected whole nodes is added. The discrepancy is easily explained. A whole should receive activation from roles only to the extent that all of its component roles are active, whereas a role should receive activation from wholes to the extent that any of the wholes it may be a member of is active.[4]

Bidirectional spread of activation allows many of the benefits of the interactive-activation model of McClelland and Rumelhart [1981] to apply in the choice of which roles and role-sets receive activation.

- The sparking of roles with parts was made to be influenced by the activations of the roles. Roles with higher activations are given higher priority in the decision of which roles to spark with a part (excluding those roles that already have a part bound to them). This is perhaps the most important optimization, and many of the other modifications were necessary so that the activation of a role would be, at all times, a good indicator that the role should be considered relevant to the gridletter being recognized. Roles with negative activation are not considered at all for sparking, so it is very important that a role receive negative activation only if it is exceedingly improbable that it is a component of the correct answer's whole.

These optimizations were performed by 1997, and led to a program that ran about twice as fast as the 1995 vintage, with a slightly higher rate of correct answers. A table comparing the performance of all three versions of the Examiner follows the

[4]The rational for the double standard is that, logically, a whole is present only if all of its associated roles are. A role is present if any one of its associated wholes is.

description of the second set of changes.

Role definitions

A second set of changes, more an overhaul than an optimization, was undertaken in 1998,[5] when it became clear that norm violations could not easily be computed using the role definitions in place at that time. The essential problem was that roles in the early Examiner were defined in ways that were highly dependent upon their exact location on the grid. Moreover, letter categories tended to have a large number of role-sets, so that when an unusual or interesting letterform was recognized by the Examiner, it was often recognized as an unremarkable member of one of the letter category's less-usual role-sets. Consequently, some unusual gridletters were often perceived by the Examiner as consisting of role-fillers that were very typical members of their role categories. A pair of examples that illustrates this problem, and its solution, can be seen in Figure 5.11.

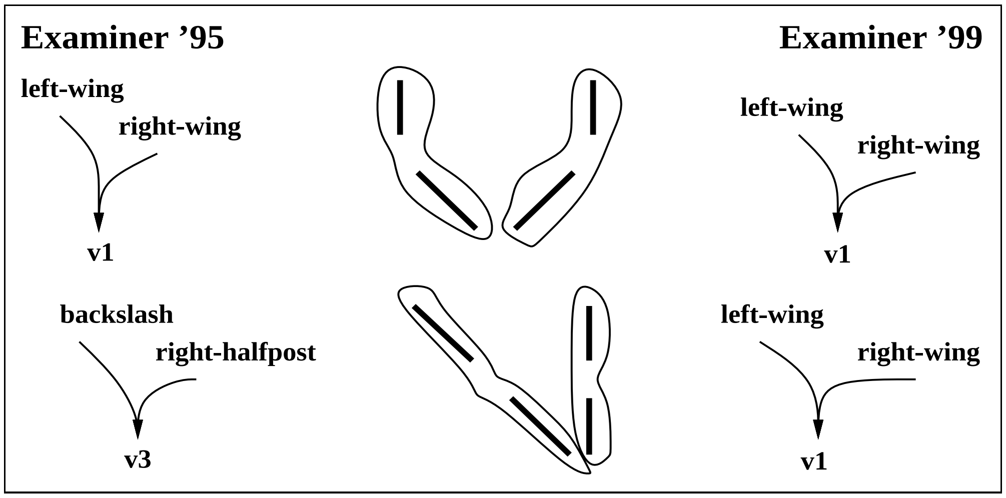

Figure 5.11: What Examiner '95 calls variety in the role-sets, Examiner '99 calls variety in the roles.

[5]The version is identified here with 1999, because the changes were completed in the early part of that year.

Two ‘v’s, very different in nature, are recognized correctly by both versions of the Examiner (the 1995 and 1999 versions). They happen to be segmented identically by both versions, as well. In the older Examiner, the definitions of the roles “left-wing” and “right-wing” are too grid-specific for the part in the lower ‘v’ to be recognized as such. The parts spark the backslash and right-halfpost roles, which are also used in role-sets for ‘x’ and ‘w’, respectively. To compensate for this specificity, three role-sets for ‘v’ existed, with this letterform corresponding to ‘v3’.

While this approach is adequate for solid Examiner performance, it fails to facilitate the work of the Adjudicator, which depends upon the parsing the Examiner hands to it. According to Examiner ’95, these two ‘v’s are both composed of highly normal role-fillers; they differ only in terms of which role-set is employed.

It is difficult to render, from that description of these ‘v’s, a general account of their styles that could be compared to, and applied to, gridletters in other letter categories. The desired solution is, as shown on the Examiner ’99 side of the figure, to parse both ‘v’s as the same role-set, with the same roles, and to note that the top one is fairly ordinary, while the bottom one has some quirks. These quirks, in the form of norm violations, can then be regarded as part of the gridletter’s style, in general terms that compare easily to other gridletters.

This case, and others like it, called for two kinds of change. One was a reduction in the number of role-sets. The second, which was necessitated by this, was a redefinition of roles in ways that are not so constrained by the grid. This required, in turn, that some new properties be added to the role definitions, as grid-specific ones were deleted. For the lower ‘v’ used in the example, the ideal would be that the parts would spark “left-wing” and “right-wing”, but *not* spark the “backslash” and “right-halfpost” roles (which still exist for ‘x’ and ‘w’). Although those parts consist of quanta that might form ideal versions of “backslash” and “right-halfpost”, in ‘x’ and ‘w’ those roles stand in very different relation to the items around them. The overhaul of role definitions included new norms such as contact and neighborhood so that, in this very example, the part on the left sparks "left-wing" (which should be touched at its *bottom* point by the other role-filler), but not "backslash" (which, in 'x', is touched in the *middle* by the other role-filler).

It was the above motivations that led to the second overhaul, but in addition, other changes, not directly related, were also incorporated. The full set of changes is described below.[6]

- The number of role-sets was drastically reduced, from 66 down to 35. The maximum number of role-sets for a letter category was reduced from four to two.

- Role definitions were made more contextual, and less dependent upon exact locations on the grid. The grid-specific labels that specified which of the twelve squares of size 1×1 a part occupied were eliminated.

- The way that the vertical and horizontal ranges of a part were represented was changed so as to allow calculation of norm violations. In the original version, the vertical range that a part spanned was expressed as a single label specifying the highest and lowest points in the part (for instance, "baseline-to-top"). The horizontal range that a part spanned was expressed by bestowing upon a part the label "left" if it touched the left edge of the grid, "center" if it touched the middle line of the grid, and "right" if it touched the right edge (obviously, many parts would merit more than one of those labels). In the overhaul, these two types of label were replaced by a general scheme that treated horizontal and vertical span in the same fashion, wherein each of the four boundaries

[6]Labels are frequently mentioned in these changes. As was explained previously, labels are not only properties of role-fillers on the grid but also are also used (in weighted lists) to define the norms of roles.

around the role-filler was considered a dimension (left-edge, right-edge, roof, and floor) with a range of possible values (three in the horizontal, and seven in the vertical). This general scheme enables easy calculation of norm violations that are abstract comparisons such as “taller”. Such comparisons are not as straightforward with labels such as “baseline-to-top”.

- The way that the tips at the ends of role-fillers were represented was expanded. Like vertical span (above), representations that combined two properties (location and orientation of the tip) were replaced by representations that separated the properties into two dimensions, with a range of values. For example, the lower tip of a “left-post” was represented in the older Examiner as one label: “down-left-baseline-tip”. In the new version, the same label was accommodated by one for location (point *5* on the grid) and one for orientation (“south”). This allows for easier comparison between cases that are alike in only one of these dimensions. The dimensions-pair representation was *added* to the role definitions, so as to augment, rather than replace, the original way of representing tips. The contrast between the two kinds of representation methods is discussed in more general terms in Chapter 10.

- Role definitions were augmented by criteria for how they should touch (have contact with) their neighbors, and how they should be surrounded by neighbors that do not necessarily touch them directly. For example, the “crossbar” role in ‘f’ and ‘t’ was originally also used for the top of a ‘z’. In the overhaul, a separate role, “z-cap” was created for the second usage. “Z-cap” differs from “crossbar” in that it should be ideally touched on its right (not in its middle), and it should have other material on the grid only below it (not below and above it).

- The number of r-role tests was reduced — from 26 to 3! Almost all of the tests in the original Examiner were highly specific to a given letter or contrast between a pair of letters. The overhaul did away with all such r-role tests. Of the three r-role tests now in the program (the functions of which have already been explained) the "filled" and "covered" tests came from the original program, and the "contact" test was added in the overhaul.

- An activity that had its own codelet type called "meta-role checking" was eliminated. This was a way of resegmenting the gridletter, based on heuristics that acted when pairs of roles, as found in one segmentation, might indicate a larger part that was involved in a different segmentation. For example, "left-halfpost" and "basebar" might be combined into "left-uparc" (the leftmost of the two roles used in 'u' or 'y'). This, like the letter-specific r-role tests, seemed excessively oriented towards the exact letters in the roman alphabet. In the overhaul, virtually all of the explicit information that is alphabet-specific is limited to the Conceptual Memory and Conceptual Network.

- A smarter means of segmentation was desirable, particularly after the elimination of the excessively specific, but helpful, meta-role checking. A routine with some top-down heuristics for parsing was added to the program; this routine is called as soon as a gestalt codelet has run, and it segments the gridletter according to a guess of the letter category of the input. The choice is made probabilistically, weighted by the gestalt value for each letter category, except in the first act of segmentation in each Examiner run, when the letter category most strongly indicated by the gestalt is guaranteed to be used to guide segmentation. The original Examiner had a bottom-up manner of initially segmenting the gridletter, based on perceptual principles like those found relevant by Palmer [1978]. The new gestalt-based segmentation routine leads more often to a correct segmentation on the first try, and many runs are accelerated considerably by this. The gestalt-based segmentation routine can also be called by other codelets along the way, when a run seems to have hit a snag.

- A small number of errors in the original Examiner were caused when two role-sets both had very high degrees of activation, which blocked either one from having the sort of dominance, in terms of activation, that leads to the end of a run. This problem is largely alleviated by the fact that there are fewer role-sets per letter category in the newer Examiner; it was thoroughly eliminated by the addition to the program of a feature that picks one role-set over the other, at random, whenever two have high positive activation. This change was made with the understanding that this condition occurs only when both role-sets seem to be perfectly acceptable answers.

- Of little interest from the standpoint of cognitive modeling, but nevertheless quite important, a radically different method of spreading activation in the Conceptual Network was implemented, speeding up that operation by more than a factor of ten. Whereas the 1997 optimization led to a decrease in the number of codelets per run, this change led to a decrease in the number of milliseconds per codelet. Additionally, the speed of graphics used by the Examiner was increased, but this is not applicable in full Letter Spirit runs, where Examiner graphics are suppressed anyway.

Many of the changes concern role definitions, some of which (namely, those resulting from the decreased specificity in terms of the grid) allowed a number of roles to be cut from the Conceptual Memory's overall inventory of roles, and some of which (namely, those resulting from the increased specificity in terms of contact and neighborhood) necessitated the creation of new roles. All told, the number of roles decreased slightly as a result of the overhaul, from 47 to 43.

Table 5.5 summarizes the performance on the "ALL" set of gridletters of the three

Examiners. The rate of correct response and the mean number of codelets per run are straightforward. The "Real speed" column shows the speed of actual runs (the reciprocal of run time) for the three versions. The 1995 Examiner's speed is used as the baseline of 1.0. These values are approximate, apply only to runs with no graphics, and exclude effects due to the programs' running on different machines (the test and development hardware for the later versions was superior to that used in 1995 and earlier).

	Correct	*Codelets*	*Real speed*
1995	82.4%	1389	1.0
1997	85.3%	430	1.8
1999	93.5%	408	18

Table 5.5: Examiner accuracy and speed increased through two phases of optimization.

The extent to which the 1999 Examiner is still, in essence, the same program as the 1995 version cannot be overstated. The paradigm of segmentation, labeling, sparking, and the spreading of activation is the core concept of the Examiner, then and now. The McGraw Examiner is the source of not only the majority of the code in the 1999 Examiner, but also a very sizable percentage of the code in the entire Letter Spirit project.

5.4.3 The module as a model of human behavior

This section provides a rough psychological interpretation of the Examiner's course of action, using the framework established in Chapter 2:[7]

[7]This interpretation puts forth the Examiner as a model of human letter recognition as it takes place in those instances where letter recognition is a slow and deliberate process involving some conscious perceptual decisions made along the way. This would take place in acts of letter recognition that take a few seconds or more (due to the stimulus being degraded in some way, such as intrinsic strangeness of the letterform). It is presumed that conscious thought is not involved in cases where

The gestalt codelet quickly assesses the gridletter's potential membership, and ladles out activation to role-set nodes in Conceptual Memory as appropriate — relatively high for role-sets that fit the gridletter's gestalt, and negative for those that do not. This models a fast-acting distributed process, which, in slow, deliberate acts of letter recognition by a person, would introduce a conscious awareness of probable candidate letter categories, and by extension, candidate role-sets. Next, the gridletter is segmented, in an interaction of top-down pressure from an active role-set and bottom-up pressure from quanta in the actual gridletter. This is accomplished in a single codelet, and the product — that of a segmented letterform — is meant to model a representation in conscious working memory. Next, looker codelets, and their successors, labeler codelets, and *their* successors, sparker codelets, endeavor to associate roles with the parts that have been hewn out of the letterform. Meanwhile, activation may spread from active role-sets to active roles and vice versa. All of these are low-level events, modeling the unconscious activity of a distributed process, though role activation can influence sparking, which is intended to represent the conscious awareness that certain roles appear to be present. The loss in activation of a highly active role-set represents the removal of this role-set from the list of possible answers in conscious consideration. In ideal runs, the remaining step is to halt when one role-set dominates all others in high activation. This models a conscious decision to prefer one role-set (and thus its corresponding letter category) over the others.[8]

In runs where the Examiner has more difficulty in finding the solution, resegmentation may occur when sparker or looker codelets are unable to do their jobs. This

recognition is very easy and takes place in a fraction of a second.

[8]It has not previously been mentioned that the Examiner gives numerical scores at the end of a run. The scores are calculated in the very last step of an Examiner run and do not influence other processing in the Examiner in any way. They are utilized by the Letter Spirit program as ratings of how well each gridletter that the Drafter generates stands as a member of its intended letter category. The score is calculated by subtracting the activation of the role-set that was returned as the answer from 100; this measure indicates strong letter-category membership with low scores.

corresponds to a conscious recognition that the current segmentation is unprofitable, and this brings a run back up to the top of the loop, though likely with a different segmentation from before.

The Examiner stands as a solid model of a particular human behavior, as it did in the original McGraw version. The real reason for implementing the Examiner, however, was to pave the way for the rest of Letter Spirit. The description of that work follows in the next three chapters.

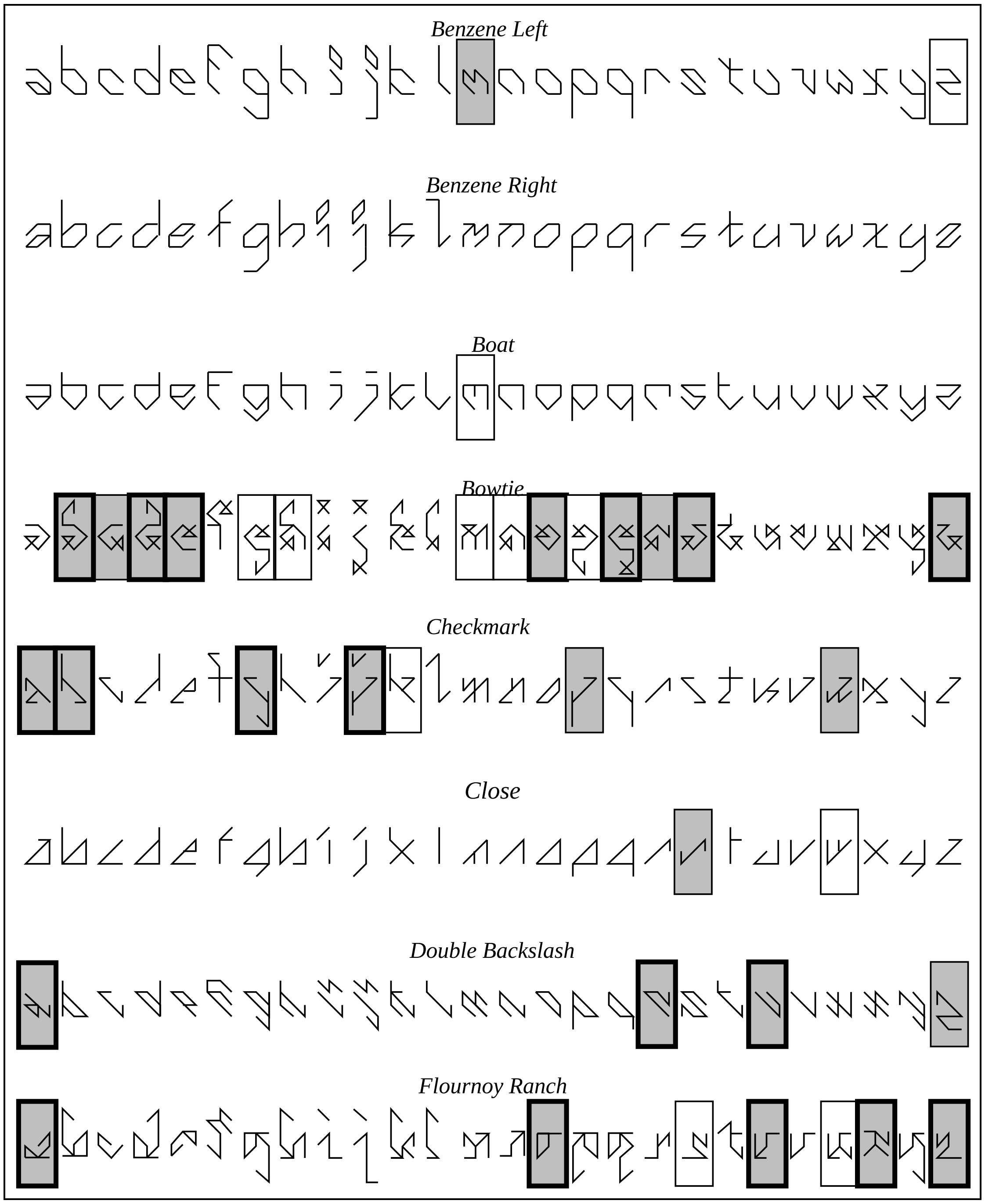

Figure 5.6: Examiner performance on the example gridfonts. The shading is explained on page 188.

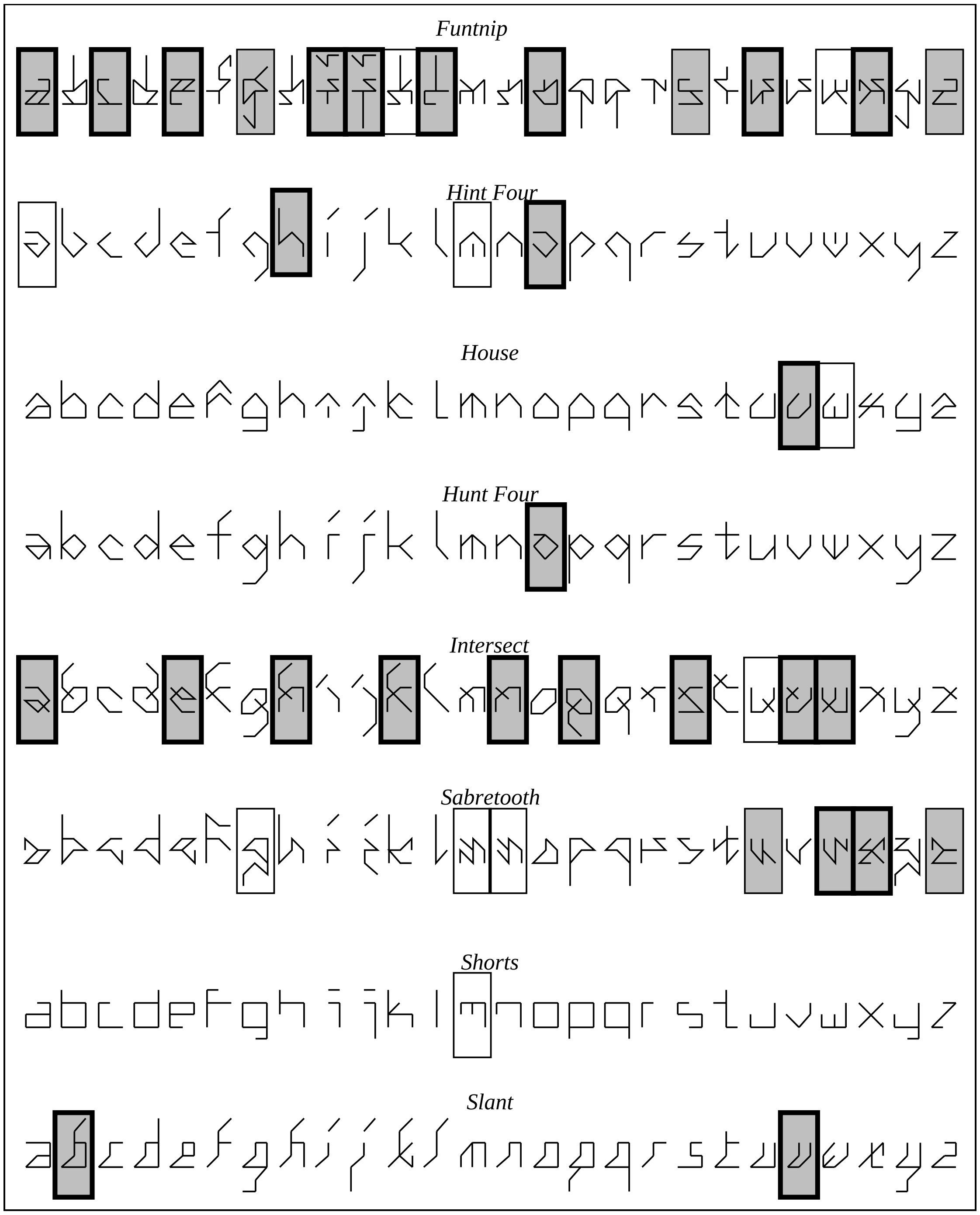

Figure 5.7: Examiner performance on the example gridfonts. The shading is explained on page 188.

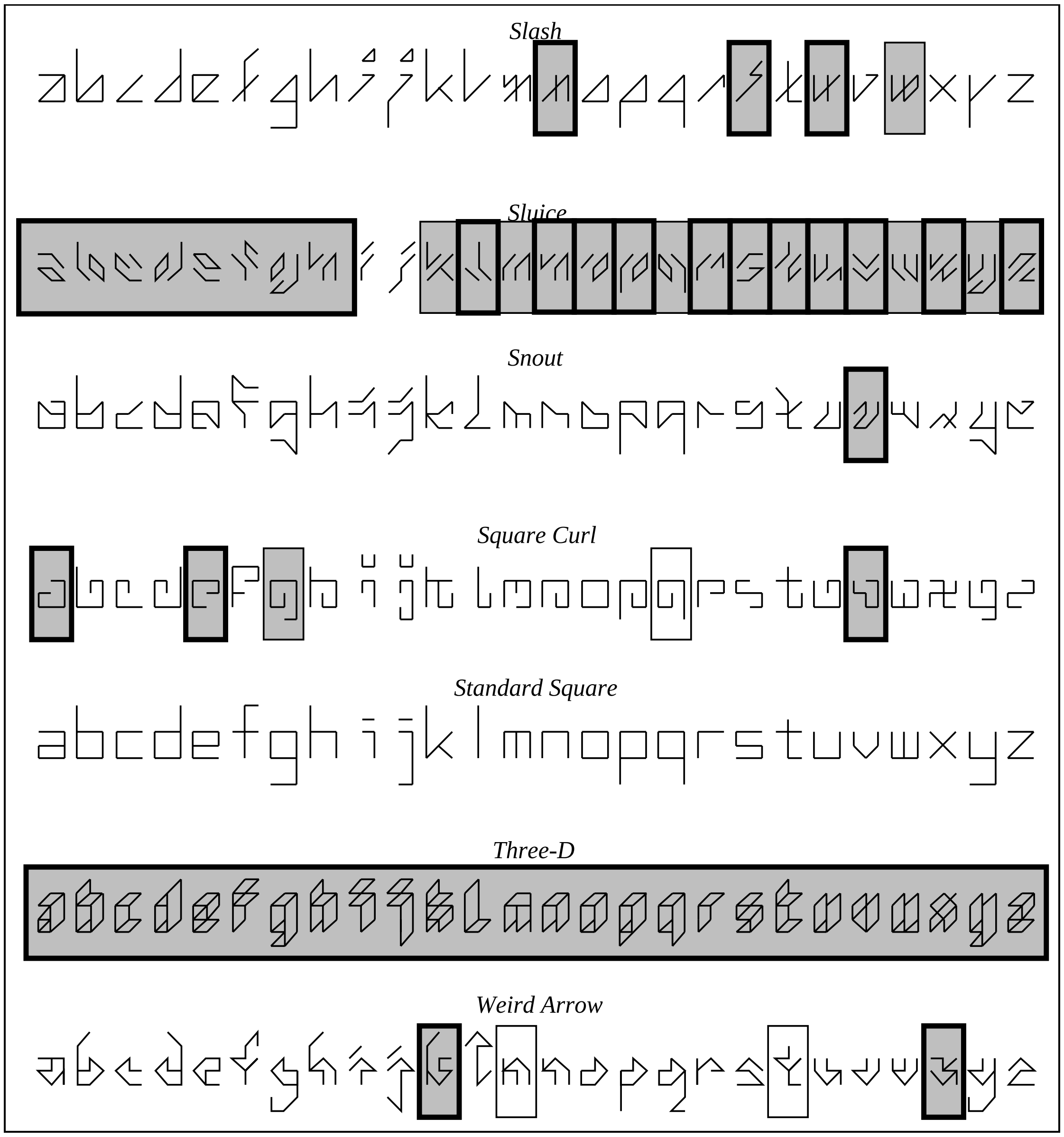

Figure 5.8: Examiner performance on the example gridfonts. The shading is explained on page 188.

CHAPTER SIX

Adjudicator

6.1 Introduction

The new work represented by this thesis centers on the implementation of two modules, the Adjudicator[1] and the Drafter. Together with the Examiner, and a modest amount of code that ties the three together, these modules complete the first implementation of the Letter Spirit program. Each of the modules employs many of the key characteristics of previous CRCC projects — namely, Copycat and Tabletop.

The Examiner, Adjudicator, and Drafter differ in ways that are due to the fundamental differences between their tasks. For example, the Drafter differs from the other two in that it models creation while they model perception. In another sense, the Adjudicator and the Drafter are alike (and the Examiner different from them) because they are both concerned with style. It is therefore in this chapter that the manner in which Letter Spirit represents style in a memory structure called the Thematic Focus is introduced. At the beginning of each gridfont-designing run, the Thematic Focus is empty. As seeds and program-designed letters are accepted (possibly tentatively) as members of the gridfont in progress, the Adjudicator fills the Thematic

[1]Many early Letter Spirit writings call this module the Abstractor. This name was quietly replaced, in-house, by "Adjudicator", around 1995.

Focus with *stylistic properties* (SPs) that cumulatively serve as a description of the gridfont's style. The importance of each SP to the style is expressed by what *level* of the Thematic Focus it is placed in, with higher levels reserved for SPs deemed to be more important to the gridfont's style.

In all, the Adjudicator performs three tasks. First, it builds in the Workspace a representation of the style of an input gridletter. Second, and concurrent with the first task, it finds ways in which the representation in the Workspace would modify the Thematic Focus, should the gridletter be accepted as a member of the gridfont. Third, it uses the degree of fit between the representation in the Workspace and the contents of the Thematic Focus to calculate a goodness rating for the gridletter.

6.2 Implementation

6.2.1 An overview of processing in the Adjudicator

The Adjudicator appears a bit stunted if compared directly to Copycat, Tabletop, or even the Examiner. One reason is that it has no likely use besides that of a follow-up to the Examiner; it cannot function in isolation of other modules, because it requires Examiner output as its input. Additionally, the Adjudicator does not have the sort of implicit loop that the aforementioned FARG programs have. That is, it does not build structure up with the potential to tear it down and start anew, progressing towards a structure it finds particularly agreeable. Instead, in each run, the Adjudicator builds up a representation of the style of one gridletter, and how it relates to the style already in the Thematic Focus. It does this in monotonic fashion, always adding to the representation and never subtracting.

Over the course of many runs on many gridletters the Adjudicator *can* both build up and tear down structure, as stylistic properties in the Thematic Focus may be either promoted (raised to a higher level) or demoted as they are found to be present or absent from the various input gridletters.[2] In the course of any single run, the program can be said to carry out three high-level steps. As with the Examiner, each of these entails the execution of many low-level codelets.

1. The Adjudicator identifies SPs in the parsed gridletter that collectively express its style. Each SP is either an abstract rule, a norm violation, or a motif (all of which were defined previously and are discussed in greater detail later). The SPs that are found in the input are stored in the Workspace.

2. Bridges, also considered part of the Adjudicator's Workspace, are built between the SPs representing the input gridletter's style (located in the Workspace) and the SPs representing the input gridfont-in-progress's style (located in the Thematic Focus). An attempt to build a bridge begins with the nondeterministic selection of an SP on one side (either the Workspace or the Thematic Focus) and a match for it is sought on the other side. Whenever a suitable match is found, a bridge is built, and the temperature is lowered a bit. The temperature is raised (and no bridge is built) when no suitable match can be found.

3. With each bridge-building success, the SP on the Thematic Focus side is put on a list of SPs recommended for possible subsequent promotion — elevation to a higher level of the Thematic Focus, which would boost the notion of the given SP's importance as a component of the gridfont's style. When an SP in the Thematic Focus is involved in a *failed* attempt to build a bridge, it is put on a list of SPs recommended for possible subsequent *demotion.* The actual promotions and demotions are not carried out by the Adjudicator. Top-level

[2]For the sake of convenience, the gridletter upon which the Adjudicator is doing its work during any given run will be called the *input.* The input may be a gridletter handed to the Adjudicator by a human, or Drafter output fed into the Adjudicator for evaluation. It is useful to have a single term for the Adjudicator's operand that applies regardless of that gridletter's origin.

control may elect to carry them out if it is decided to keep the input as a member of the gridfont.

Unlike the Examiner, the Adjudicator runs through its high-level steps in a fixed, predetermined order. All SPs are found before any bridges are built and bridge-building is completed before the lists of potential promotions and demotions are compiled. Each step is guaranteed to take place exactly once per run, always in the same order, and then the Adjudicator run is over. However, within any Letter Spirit run, the Adjudicator must run many times, and it is over the set of these many runs that the process of creating the representation of a style can be quite complex.

6.2.2 Memory structures in the Adjudicator

Workspace

The Adjudicator's Workspace begins with the Examiner's output: a parsed gridletter, which means the Examiner's judgment of the gridletter's letter category plus a breakdown into parts identified in terms of the roles that they are meant to represent. The Adjudicator then adds to the Workspace a repository of all three types of SP — motifs, abstract rules, and norm violations — that it finds in the input. A second component of the Adjudicator's Workspace is the set of bridges that link SPs in the Workspace with corresponding (similar or identical) SPs in the Thematic Focus. The third part of the Workspace is the set of recommended promotions and demotions. The Adjudicator merely creates this list; whether or not these promotions are actually carried out is determined by Letter Spirit's top-level loop.

Thematic Focus

Activity in all portions of Letter Spirit other than the Examiner revolves around the Thematic Focus. This data structure, shared by the Adjudicator and the Drafter, represents, at any given time, the program's conception of the style of all those gridletters that have been accepted as members of the gridfont under development. The Adjudicator both reads from and writes to the Thematic Focus, while the Drafter only reads the Thematic Focus without changing it.

The basic design is simple. Essentially, the Thematic Focus is a two-dimensional matrix that stores stylistic properties, the "atoms" of style, that have been found to be distinguishing characteristics of the gridfont in progress. Each column holds a different type of stylistic property (norm violation, motif, or abstract rule — all defined in Chapter 1). Each of the six rows, or *levels*, in the Thematic Focus expresses the degree to which the SPs contained in that row are important to the style of the gridfont. Promotion (and demotion) of stylistic properties throughout a run is probabilistic, so it is not possible to say exactly what frequency of occurrence in the gridfont each level corresponds to. The lowest level is essentially the repository of stylistic properties that have been found in only one gridletter, and the next level, those that have been found in exactly two gridletters. Table 6.1 describes the approximate degree of importance associated with each level in the Thematic Focus. Because promotion is handled nondeterministically, these descriptions are approximate. What is almost always true of promotions is that as the Thematic Focus acquires content, the SPs that are most important to the style move to the top while less important ones remain on lower levels.

Levels of Enforcement

Initially, the Thematic Focus is empty. When the first gridletter (a seed, if Letter Spirit is given any) is accepted as a member of the gridfont, all of the SPs detected when the Adjudicator ran on the gridletter are added to the Letter Rows level of the Thematic Focus. When a new gridletter is added to the gridfont, its SPs are promoted probabilistically. Any SP already in the Thematic Focus and also in the new gridletter

Level	*Found in how many letters*
Near-Universal SPs	All or almost all
Frequent SPs	A clear majority
Common SPs	About half
Occasional SPs	Three or more
Two-Time SPs	Exactly two (usually)
Letter Rows	Exactly one (usually)

Table 6.1: Levels of the Thematic Focus.

is likely to be moved up to a higher level. Any SP already in the Thematic Focus but not detected in the new gridletter may be demoted. Any SPs not yet in the Thematic Focus present in the new gridletter are added to the Letter Rows level. Thus, each level represents approximately the number and proportion of gridletters in the gridfont in which the SP has been detected. Letter Rows indicates that an SP has (probably) been detected exactly once. Two-time SPs are those SPs detected (probably) twice. Near-universal SPs are usually those SPs that are in all or nearly all of the gridletters in the gridfont. The intermediate levels represent approximate gradations along the scale between two occurrences and universal occurrence.[3]

The structural division of the Thematic Focus into levels helps realize the principle of "Levels of Enforcement" — the idea that while there may be many attributes to a style, they should not all be treated as though they were equally important to the style. The level to which an SP has risen in the Thematic Focus indicates (approximately) its prevalence in the seed gridletters. This factor, along with the *type* of SP — and in the case of motifs, the *size* of the motif — determines how much importance to give the SP in Adjudicator and Drafter calculations.

Level of Enforcement is used in two different ways to determine how much influence

[3]An SP can only rise at most one level in the Thematic Focus each time it is detected in a letter. Therefore, when the number of gridletters added to the gridfont is less than six, an SP that is present in each of them can still only be that many levels high in the Thematic Focus. For example, if early in one Letter Spirit run only four gridletters have been added to the gridfont, then an SP that was detected in all four of them would probably be in the Common SPs row (that is, the fourth row).

a given SP has upon processing in the Adjudicator and Drafter. First, the chance that an SP is chosen to exert an influence upon processing is tied to the level of the SP in the Thematic Focus. The full details will be given later, but for now, it is sufficient to say that the influence of the Thematic Focus, as a whole, upon processing takes place by many individual actions, each involving a single SP from the Thematic Focus. The selection process is nondeterministic, but weighted by level, so that SPs higher in the Thematic Focus have a greater chance of being selected than those in lower levels. Second, when an SP is used to influence processing, the amount of influence is multiplied by a factor giving more influence to those SPs in higher levels; as was noted earlier, type and size of SP are also factors in those calculations.

As a result, SPs that have previously proven to be important (that is, SPs that have occurred frequently in the seeds) have a high degree of influence on subsequent decisions. This creates a strong pressure towards coherence as new gridletters are created and evaluated for style.

Growth of the Thematic Focus

The Thematic Focus does not grow very much as a result of any single run of the Adjudicator. Rather, many Adjudicator runs, over the course of a full Letter Spirit run, gradually build up the contents of the Thematic Focus, and thereby the program's representation of a gridfont's style.

If the number of SPs that might possibly be found in any gridletter is N (N will, of course, be quite large) and if the mean probability of a given SP being present in a given gridletter is $\frac{1}{k}$, then the number of SPs that are found in every one of n gridletters will be $\frac{N}{k^n}$. For large k, the number of SPs found in $n-1$ of the gridletters will be approximately k times greater. Therefore, the Thematic Focus will tend to take on a pyramidal shape, with many items in the lowest levels, and few in the highest.

The assumption that the distribution of SPs in a set of gridletters is random is an oversimplification, particularly when the gridletters have been designed to share a style! However, the basic pyramidal shape does tend to occur. Figure 6.1 shows the median number of items in each level of the Thematic Focus for styles based upon five seed gridletters. The sixth and highest level of the Thematic Focus is necessarily empty after exposure to five seeds, because an SP can be promoted upward at most one level per seed. Figure 6.1 is based on a set of Letter Spirit runs reported at greater length in Chapter 8. The numbers in the figure are based on data from runs on what will henceforth be called the SMALL set of gridfonts: Benzene Right, House, Shorts, Snout, and Standard Square, each time starting from seeds from a set of letter categories that will henceforth be call the BCEFG set (consisting, not surprisingly, of ‘b’, ‘c’, ‘e’, ‘f’, and ‘g’).

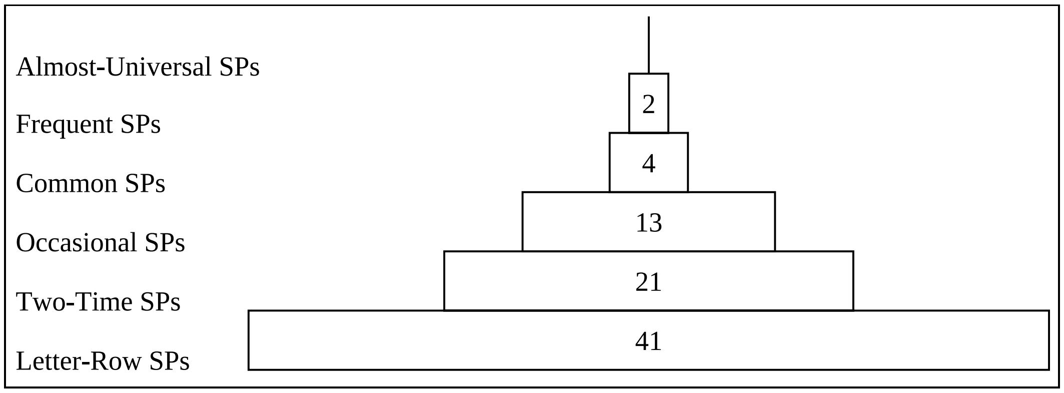

Figure 6.1: The number of items per level in the Thematic Focus tends to create a pyramidal shape.

The Adjudicator’s handling of promotion and demotion of SPs, and of their use in influencing subsequent processing is intended to appropriately reflect the relative importance of an SP, based — in part — upon the SP’s prevalence in the gridfont. Promotion and demotion in the Thematic Focus occur nondeterministically, so there is not an exact correspondence between a level in the Thematic Focus and the frequency of the SPs it contains. An SP can be expected to be promoted (moved up one level) each time it is found in a seed, and demoted (moved down one level) roughly half the number of times that it is not found in a seed. In terms of influence, an SP's effect upon Adjudicator scores and on how much it influences drafting (see Chapter 7) is doubled each time it rises one level in the Thematic Focus.

In terms of cognitive modeling, the organization of the Thematic Focus into levels is a way to get at the distinction between items in working memory that will attract attention at various times throughout a run and those that will not. SPs that are high in the Thematic Focus are those that will tend to attract more attention than those in lower levels. Of course, no one has ever suggested that attention, in people, is realized by means of six discrete, ordered ranks. How well the Adjudicator succeeds in its tasks may provide some indication of how well the Thematic Focus's organization into levels serves as a model of human attention.

Library

When a gridletter is used as a part of the basis for the program's representation of style, the SPs that are attributes of the gridletter and its role-fillers are stored (or promoted) in the Thematic Focus, while the gridletter and its role-fillers are themselves stored in the *Library*. The structure of the Library is quite simple; for each letter category, it contains the gridletter (if any) that has been accepted as the gridfont's best version of that category, as well as the gridletter's role-fillers, labeled by role name. The Library is important for a certain mode of the Drafter that creates new gridletters by borrowing shapes from other ones. It also has a modest effect on the calculation of temperature in the Adjudicator (with role-fillers in the Library basically being treated as motifs pertaining to the gridletter). The Library does not have any other impact on the workings of the Adjudicator, and so it will not be mentioned again until Chapter 7.

Conceptual Memory: Stylistic properties

The Adjudicator is concerned primarily with style, but it must also know about letters. To calculate norm violations, in particular, it must know what the norms of letters are if it is to determine when they have been violated. Therefore, the Adjudicator accesses the same role and role-set definitions, stored in the Conceptual Memory, that the Examiner uses in recognition. In addition, the implementation of the three types of stylistic properties (SPs) that are used by the Adjudicator and the Drafter is part of the Conceptual Memory.

Motifs are perhaps the most pervasive of the three types of SPs, being found in almost every gridfont. The number of motifs that might potentially exist in the set of all possible gridfonts is virtually boundless, so they are defined in terms of routines for finding motifs, rather than a comprehensive list of all motifs possible. Motifs come in five varieties; all of them are in some sense *shapes*, but the five varieties distinguish varying amounts of generality. The least general (the most literal, one might say) are *literal motifs*. A literal motif is a specific shape that must occur in a specific position on the grid, and in a specific orientation; anything that deviates from this is not considered the same motif. *Translatable motifs* are a bit more general, holding shape and orientation as fixed, but allowing position to vary. A translatable motif is a shape on the grid that can be slid around horizontally and vertically, but maintains a fixed orientation. *Turn-180 motifs* allow a shape to be translated, reflected vertically or horizontally, rotated 180°, or any combination of those transformations. *Turn-90* and *Turn-45* motifs are more general still, allowing all the transformations that apply to Turn-180 motifs, plus rotations in multiples of 90° and 45°, respectively. In the case of Turn-45 motifs, it is interesting to note that rotation by any angle that is a multiple of 45° but not a multiple of 90° changes the lengths of all the constituent quanta, toggling quantum length between 1 and $\sqrt{2}$, turning vertical and horizontal quanta into diagonal quanta, and diagonal quanta into vertical and horizontal ones.

The Adjudicator finds motifs of these types by noticing whatever shapes happen to be in the input, and matching these against motifs already in the Thematic Focus. This is currently the only type of SP in Letter Spirit that is richly productive — for which the number of potential instances is truly vast, and for which Letter Spirit has a set of routines for defining the instances, rather than simply a set of possible instances.

There is a fixed set of 30 abstract rules built into Letter Spirit. Precisely half of these are expressed in terms of bans on zones in the grid; a gridletter conforms to one of these abstract rules when it does not contain any quanta that are inside the given zone. In addition, six abstract rules ban particular orientations of quanta, and six more ban particular angles between pairs of touching quanta. Two more ban straight, collinear stretches of segments longer than some maximum length (namely, 2 quanta and 3 quanta, for the two rules respectively), and one more bans straight, collinear stretches of segments *shorter* than 2 quanta long. Theoretically, the program could be given the means to generate new abstract rules, as it does with motifs, but that possibility is left to future work. Finally, in addition to the definitions of the abstract rules, the Conceptual Memory also has a list, for every letter category, of which abstract rules are worth noting for that letter category. This prevents the program from foolishly inferring, for example, that the abstract rule that would ban all descenders should apply to a gridfont based only on its having seen several gridletters that belong to categories that normally lack descenders anyway.

Finally, norm violations make up the third type of stylistic property. There are several types of norm violation. Most of them are defined in terms of a dimension along which a role-filler may be measured, and the norm violation notes whether this quantity, in the role-filler, is different from the value expected in the norms for the corresponding role. Eight norm-violation types are of this kind, and the dimensions they are concerned with are height, width, weight (the total number of quanta in a part), curvature of the role-filler, and the locations of its maxima in the four directions left, right, up, and down. Another type of norm violation notes how role-fillers' contact with their neighbors compares with the norms for their corresponding role; for example, when the norm is for a role's filler to be touched at the tip, but it is instead touched somewhere in the middle, a norm violation "touch-inward" is noted. The final type of norm violation notices the direction that tips may be shifted from their norms; for example, "tip-south", "tip-east", etc. Norm violations are mildly productive; each type here is, in a sense, two types, because the Adjudicator records both *relative* norm violations, expressed in terms of the difference between the norm and the value actually detected (for example, "wider") and *literal* norm violations, expressed in terms of the exact mapping between the expectation set up by the norm and the value that was actually detected (for example, "skinny→wide"). In all, several dozen types of norm violation exist, although the variety of norm violations perceivable by Letter Spirit does not compare to the rich variety of norm violations that people can notice spontaneously while considering letters.

Conceptual Network

The Examiner features a localist connectionist network with nodes representing role and role-set concepts. During a run, activation in this "Conceptual Network" has an important influence on other activity during a run. The extension of this network to include stylistic concepts of importance to the Adjudicator (and Drafter) was considered, but ultimately not implemented. Chapter 9 discusses the merits of making such an extension part of future work on the program.

Coderack

In the Examiner, different phases involve the same kind of processing, and differ only in the values of a couple of parameters that change over the course of a run. For the

Adjudicator, in contrast, a run consists of three highly heterogeneous phases. Each of the three distinct phases calls for the Coderack to be initialized with a large number of codelets. Unlike the Examiner, whose codelets post additional codelets in ways that can make the lengths of runs vary widely, the Adjudicator posts new codelets during a run only in predictable ways. An Adjudicator run simply drains the Coderack dry, one codelet at a time, and when it is empty, the phase is over, and the next phase begins with the posting, in a lump sum, of most of the codelets that will run in the next phase.

The way in which codelets are posted to the Coderack in an Adjudicator run can be completely described as follows: In the first phase, the Adjudicator posts 600 comparers, 500 stenographers, 100 constables, and 400 worms. All of these codelets carry out the same kind of function — they all detect SPs (with one codelet type for each type of SP). In addition, each first-phase codelet may (with 50% probability) post a replacement for itself — a codelet of the same type but that looks for a different specific SP. This, in effect, makes the number of codelets of each of those kinds that will actually run about double the number that is originally posted. The number of first-phase codelets that will run is therefore largely predictable, varying just a bit from run to run. No codelets in the other two phases, however, can post any new codelets. The second phase posts 150 of each of the two types of bridge-building codelets — bridge-aheads and bridge-backs (although when only one gridletter's SPs have been added to the Thematic Focus, only 100 of each type are added in this phase). The third phase is highly homogeneous, with 150 promoters posted, each of which handles one item on the list of bridges created in the second phase. This phase, and the Adjudicator run as a whole, ends when either the bridge list or the Coderack is empty. The Adjudicator codelet types will be summarized below.

Temperature

Temperature is not used in the Adjudicator in all the ways that it is in the Examiner, where it helps to determine how codelets are run off of the Coderack, can be used to determine whether or not a run should end, and is used in the calculation of the Examiner's rating of how well the input serves as a member of the answer's letter category. In the Adjudicator, each phase consists of very similar kinds of codelets. As a consequence, they all have equal urgencies, and therefore temperature would not have any effect in determining to what extent urgency influences codelet selection. In addition, the end of an Adjudicator run (or phase) is determined exclusively by when the Coderack is empty; therefore, temperature is not used in that way, either. Temperature is essentially nothing but a scoreboard that is used to rate how well a gridletter matches the previous contents of the Thematic Focus in terms of style. During the second phase — the bridge-building phase — of an Adjudicator run whenever an SP is found in both the Workspace and the Thematic Focus, the temperature is lowered a small amount (depending upon the rating of how important the SP is). When an SP is found in the Workspace or the Thematic Focus, but not both, temperature is raised. Therefore, low temperature means that the input is a good match for the gridfont's style.

Successful bridge-building indicates that the Adjudicator is successfully relating the input to the style that had previously been built up during the Letter Spirit run, and therefore causes the Adjudicator to lower the temperature. Failure in bridge-building leads to the temperature being raised. More specifically, when an SP is found in both the Workspace and the Thematic Focus, whether by a bridge-forward codelet or by a bridge-back codelet, the temperature is adjusted a small amount towards zero, to reflect the fact that useful structure is being built up. When an SP is found in the Workspace but not the Thematic Focus, temperature is adjusted towards the upper middle of the scale (typically, near 75). When an SP is found in the Thematic Focus but not in the Workspace, the punishment is a bit more severe, and the temperature is adjusted towards 100. In all cases, the magnitude of the change made to the temperature depends upon the type of SP, and some additional manipulation takes place to ensure that the temperature is always in the range from 0 to +100.

6.2.3 Codelets

A summary of all codelet types in the Adjudicator, separated into the three phases, is offered below:

First Phase

comparer codelet (arguments: part, trait dimension)

The part will already have been identified by the Examiner as a role-filler for some role. The part and its corresponding role are compared in terms of the specified trait dimension (for example, height), and the difference between them (a norm violation unless there is no difference), expressed in relative terms (for example, "taller"), is added to the Workspace.

There is then a one-half probability that a new comparer codelet will be posted with randomly selected arguments.

stenographer codelet (arguments: part, trait dimension)

The part will have already been identified by the Examiner as a role-filler for some role. The part and its corresponding role are compared in terms of the specified trait dimension (for example, height), and the exact difference (a norm violation unless there is no difference) between them (for example, "short→tall"), is added to the Workspace.

There is then a one-half probability that a new stenographer codelet will be posted with randomly-selected arguments.

constable codelet (argument: abstract rule)

If the input obeys the abstract rule (for example, if the rule is "ban diagonal quanta" and the input contains no diagonal quanta), then the rule is added to the Workspace.

There is then a one-half probability that a new constable codelet will be posted with a randomly selected abstract rule as its argument.

worm codelet (argument: motif-type)

This codelet finds a shape that is part of the input by means of a "worm". The worm begins at a randomly-selected point in the input and crawls randomly along its quanta for a while (the worm halts at tips, when it has completed a loop, or after a nondeterministically chosen number of steps). The shape specified by the worm's path is added to the Workspace as a motif of motif-type.

There is then a one-half probability that a new worm codelet will be posted with a randomly selected motif-type as its argument.

Second Phase

bridge-back codelet (argument: Thematic Focus level)

A stylistic property is selected at random from the given level of the Thematic Focus. If there is a match for that stylistic property in the Workspace (such that the one in the Workspace is identical to or more specific than the one in the Thematic Focus), then a bridge is created between the Thematic Focus and Workspace linking the two stylistic properties, and the temperature is nudged a bit in the direction of 0. If there is no match, then the temperature is nudged a bit in the direction of 100.

bridge-ahead codelet (no arguments)

A stylistic property is selected at random from the Workspace. If there is a match for that stylistic property in the Thematic Focus (such that the one in the Workspace is identical to or more specific than the one in the Thematic Focus), then a bridge is created between the Thematic Focus and Workspace linking the two stylistic properties, and the temperature is nudged a bit in the direction of 0. If there is no match, then the temperature is nudged a bit in the direction of a target temperature (roughly 60, but varying according to a few factors).

Third Phase

promoter codelet (no arguments)

A bridge is selected at random and added to the promotions list in the Workspace.

6.2.4 Parallel terraced scan

Despite the fact that a single run of the Adjudicator is essentially a feed-forward process, it, like the Examiner, taps into the power of the parallel terraced scan.

The first way in which the Adjudicator realizes the parallel terraced scan is part of the design of the bridge-building phase of a run. In this phase, bridge-ahead codelets take the SPs that were found in the input gridletter in the first phase, and look for matches for them in the Thematic Focus. Consequently, processing focuses on those SPs that seem most promising as part of the gridfont's style.

The second way that the Adjudicator employs the parallel terraced scan is in biasing its nondeterministic selection of SPs, at various places in the code, so as to select those found on higher levels more often. This also focuses processing on the SPs that are liable to be most important.

Finally, a third way that the Adjudicator realizes the parallel terraced scan is one that cannot be seen during any single run, but occurs over multiple runs. The Thematic Focus stores SPs that occur in gridletters that the Adjudicator reviews. As was noted earlier, a pyramidal shape naturally emerges as a style is built up from seeds. This indicates that processing has focused preferentially on those SPs that are most common in the gridfont and therefore most important to the gridfont's style, and focusing processing on what is most important is the essence of the parallel terraced scan. It is fair to say that this shape represents emergent behavior of the Letter Spirit program as a whole rather than of the Adjudicator alone. Chapter 8 describes the ways that Letter Spirit's top-level control can carry out the parallel terraced scan.

6.3 Performance

6.3.1 The Adjudicator's tasks

In contrast to the Examiner, the Adjudicator has a bewilderingly large number of tasks, all of which are interrelated and are accomplished by means of much the same activity, but are quite distinct in terms of the actual product. The Adjudicator finds the SPs in the input. It rates the input according to how well it represents the goal style for the current run as built up thus far. It specifies how the Thematic Focus should be updated to integrate the input into its notion of the goal style. Over the course of many runs on individual gridletters, it builds up a style. The Adjudicator can be tested on how well it does any of these tasks, but one of the most important tests of the Adjudicator's performance is how well the Thematic Focus that it builds allows the Drafter to produce new gridletters in the goal style.

6.3.2 Representing a style

Using a set of gridletters to build a representation of those gridletters' aggregate style is the essential task of the Adjudicator; that this be done well is vital to all Letter Spirit activity involving style. It has already been explained what the approach is intended to achieve in terms of representing style, but viewing the contents of the Thematic Focus after it has captured the style of a few gridletters provides insight into how, and how well, the approach works in practice. Here, the styles of the SMALL set of gridfonts, based on the BCEFG training set, are used to illustrate Adjudicator behavior.

The full contents of the Thematic Focus can be a large — even overwhelming — agglomeration of data. The sheer amount of information, even for these very basic examples, makes a comprehensive display of the Thematic Focus's contents unwieldy. The number of SPs in the Thematic Focus, for any given one of these examples, ranges from 76 to 110. To make this tractable, only the SPs in the highest levels are displayed, followed by the number of SPs in the lower levels. With only five seeds, the highest level, Near-universal SPs, is always empty. In the displays here, the full contents of the next two levels (Frequent SPs and Common SPs) are shown, followed by the number of SPs in each of the three remaining levels. In terms of influence, the highest levels always dominate anyhow, so the details in the unshown lower levels tend to have at most a minor functional role in any activity involving the Thematic Focus. Figure 6.2 and Figure 6.3 show Thematic Focus contents for the five gridfonts in SMALL.

The most obvious property of these displays is the dominance of motifs over the other kinds of SPs. In the Frequent SP level, for all of the styles except Standard Square, motifs heavily outnumber the other kinds of SPs combined. In fact, lumping abstract rules and norm violations together, the other four styles average just one SP apiece, through both the top two displayed levels. This may be taken as a sign that the Adjudicator is heavily reliant on motifs, but it is equally valid to note that the styles mentioned are heavily reliant upon motifs. The first three were chosen as part of the SMALL set of gridfont as examples of motif-based styles with high recognizability. It is natural that motifs are the way that these styles are represented.

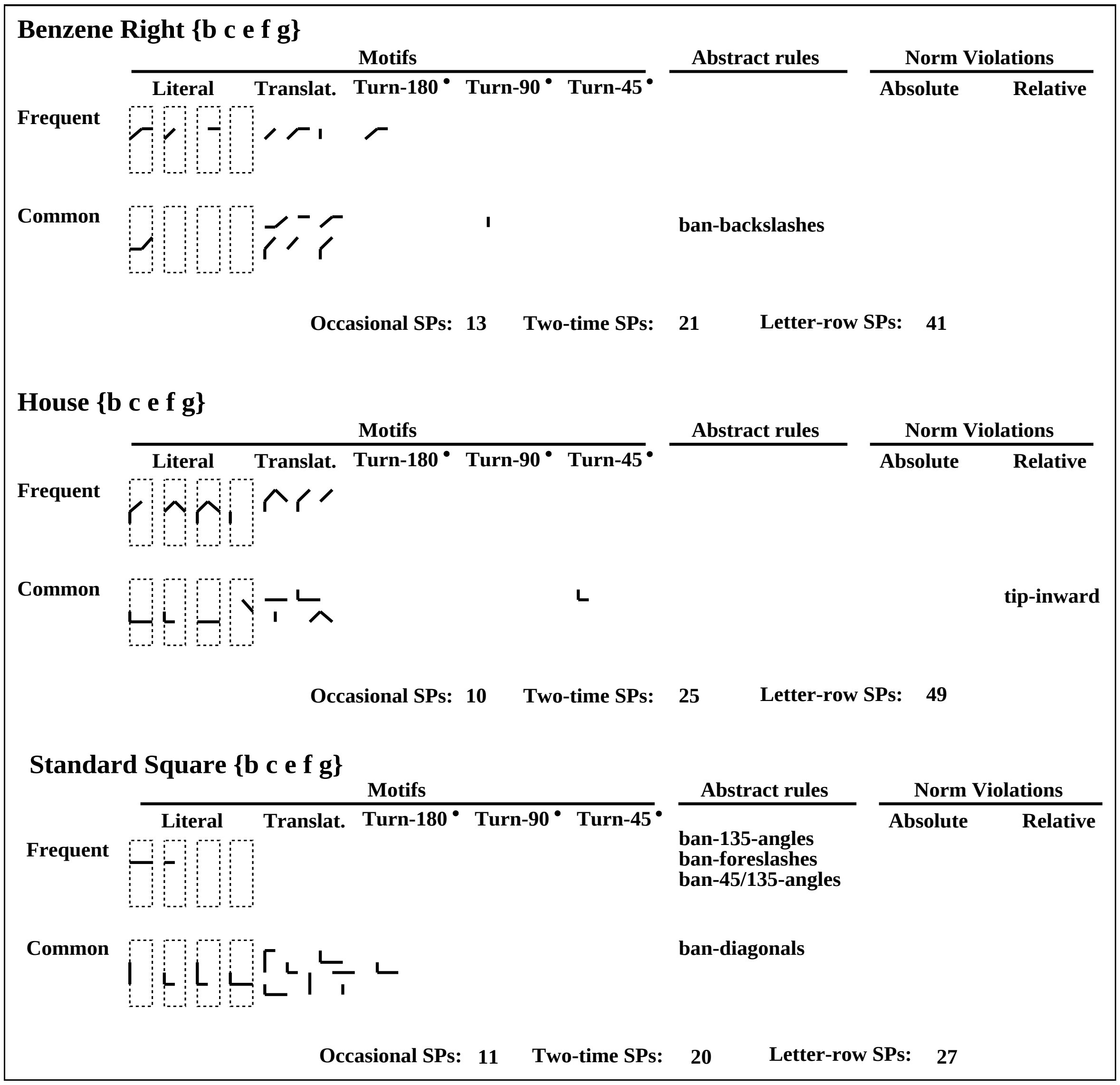

Figure 6.2: Thematic Focus contents: Three gridfonts.

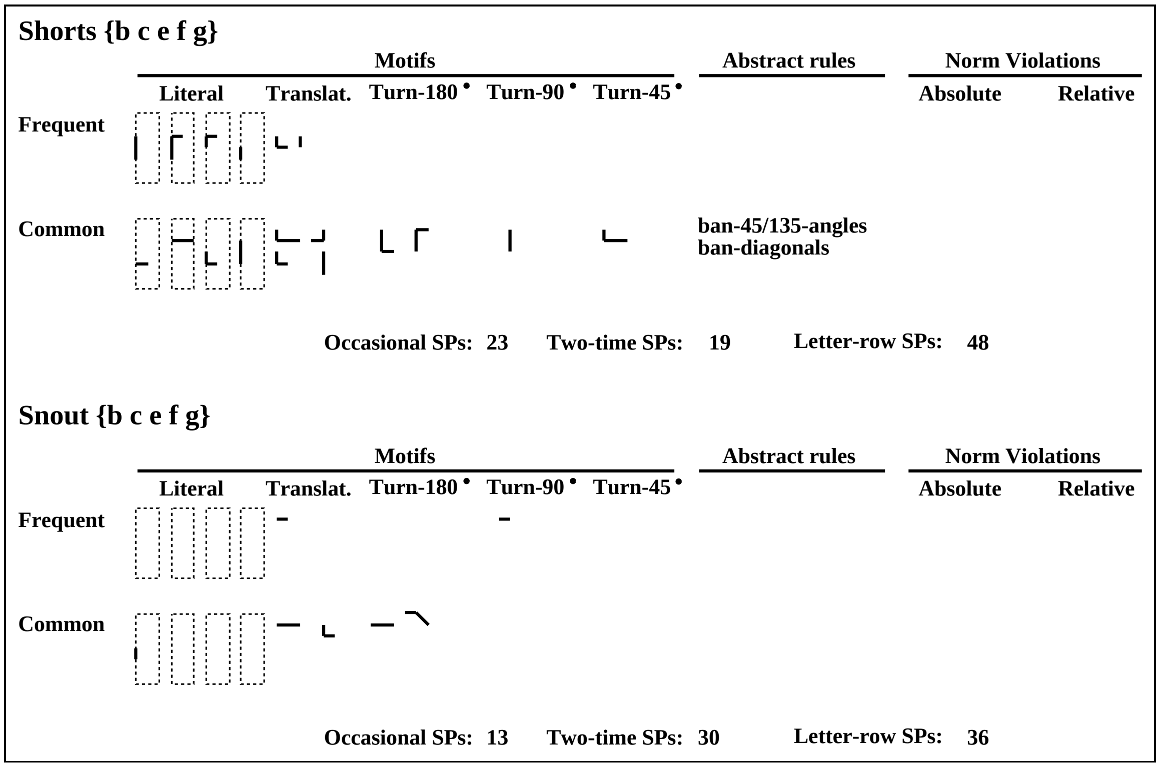

Figure 6.3: Thematic Focus contents: Two more gridfonts.

The other two gridfonts were included in SMALL to capture variety, with gridfonts that are not defined so strongly in terms of an o-ring motif. Shorts is similar to Standard Square in that it is defined in part by the same squarish motifs, and the same abstract rules prohibiting diagonality. Shorts, in fact, is essentially Standard Square with the significant addition of the tendency for parts to be shorter (not necessarily only in the vertical direction) than normal. This trait, however, does not show up in any obvious way in the Thematic Focus. The reason is that shortness is not one of the norm violations that are built into Letter Spirit. It is, nonetheless, represented indirectly. Any particular way that a role-filler is drawn shorter than its norm would call for is some kind of norm violation already built into Letter Spirit ("weight lighter", "height shorter", "tip inward", etc.). A set of seeds from Shorts places all of these norm violations into the Thematic Focus, and so the concept of shortness is implicit in the presence of all of these kinds of shortness. This is not fully effective, however, because none of the kinds of shortness is all that common. Shorter height, for example, is present only in one of the five BCEFG seeds for Shorts. As a result, all of these properties are present in the Thematic Focus, but on rather low levels, where their influence is quite modest (and which excludes them from the display in the figure, though they *are* in there, in the two lowest levels). For styles, like anything else that is complex, the whole is greater than the sum of its parts. The way that the Adjudicator fails to represent Shorts is an excellent example. The actual style of "shortness" that should be on the highest level of the Thematic Focus is represented by a few (say, four) substyles, perhaps three levels below the top. With the way influence is calculated in the Thematic Focus, this amounts to at best four-eighths (four being the number of substyles, over two to the number of levels lower in the Thematic Focus that the substyles were found than where they ought to have been), or one half, of the influence that the real style *ought* to have. A proper representation of Shorts would require the program to have a built-in norm violation for shortness that was the superset of shorter height, narrower width, etc., or else would be a way for the program to dynamically generate new types of norm violations, after noticing on its own that shorter height, narrower width, and other norm violations were all fundamentally related. This is beyond the program at present, and performance on Shorts is adversely affected as a result.

Finally, Snout is based upon a motif that rotates, and, ideally, its style would be represented in the Thematic Focus with one or more motifs of the rotatable types. Indeed, one of the Common SPs that is noted is a two-quantum Turn-180 motif with a $135°$ angle. This is only a partial success, however. The characteristic motif of Snout does include such an angle, but it is a larger shape than that. The fact that a third, additional quantum that is adjacent to this two-quantum shape in four out of five of the given seeds was not recognized is detrimental to performance. It is not a fluke that the motif that is found is smaller than what shape is actually common to all or most of the seeds; in fact, the Adjudicator did not place a motif of more than three quanta in the highest levels of the Thematic Focus for *any* of the gridfonts in this test set, including the three that are strongly motif-based. The major difficulty is that the number of potential motifs is so large that exhaustive search is impossible, and blind search misses some that are important. Nonetheless, motifs common to the seeds are found, and for Snout, in this example, the aforementioned two-quantum motif made it to the level of Common SPs. The Adjudicator does not do a very good job of pinpointing the style of Snout, however. This motif can be found in *all* gridletters with a 135° angle, but not all such gridletters include the full snout shape. In addition, this motif does not capture the *salience* (recall Chapter 1) of the snout motif, which always points outward and violates the flow of the gridletter a bit. These concepts are not part of the current implementation, and so performance on gridfonts like Snout is compromised.

None of these gridfonts provide an example of a style that was characterized by the Adjudicator *primarily* in terms of abstract rules or norm violations, but this is because all of them actually did have a high degree of their spirit couched in terms of shapes. As the cases of Standard Square and Benzene Right illustrate, abstract rules *can* be detected effectively when they apply. The norm violation "tip inward" in the Thematic Focus built up from House is an astute observation on the Adjudicator's part, which shows its ability to emphasize that type of stylistic property when it suits the situation.

Although these representations of styles are somewhat successful, they demonstrate one major shortcoming that impairs the Adjudicator's overall success, and the problems that have already been noted are just instances of that general shortcoming: when a style has a single defining characteristic that would be obvious to a person, this is rarely noticed as such by the Adjudicator and placed in the highest applicable level of the Thematic Focus. The organization of the Thematic Focus into levels is intended to collect the most important stylistic properties at the top, but this rarely occurs. The reasons are many. Some stylistic properties are beyond the Adjudicator's ability to represent, because it does not define SPs in a way that is richly productive. The Adjudicator cannot produce a new SP by modifying an existing one with context, salience, or flow considerations (and those were all shown in Chapter 1 to be important). In contrast, o-ring motifs, which are perhaps the best way to express the styles of Benzene Right, House, and Standard Square *can* be represented by the Adjudicator, but do not make it to the top of the Thematic Focus. This is because promotion in the Thematic Focus is determined by how widespread an SP is among the seeds, and o-ring motifs occur (typically) in only 6 of 26 gridletters (in the BCEFG seeds used in the tests, two of the five). This specific problem might be fixed by building in a sense of how often a stylistic property occurs in opportunities where it has a chance to, as was implemented already for abstract rules. The failure of 'c', for example, to feature an entire o-ring should not be a surprise, and should not penalize the level in which an o-ring motif is stored. The problem is in specifying which SPs can be pardoned for not occurring in which letters. For a fixed set of SPs, like the special case of the o-ring, or for abstract rules as they are currently implemented, this is tractable. However, for a productive class of SPs, like motifs, there is no obvious general solution. At present, the problem appears to be fairly pervasive in light of the fact that it can be caused by both having SPs that are *not* productive and by having SPs that *are* productive! Any solution to this deep and general problem in Letter Spirit would have to be of considerable sophistication.

The examples of Thematic Focus contents provide insights into how the Adjudicator represents style that could not be captured with numbers alone. Nonetheless, performance in terms of accuracy provides other insights, and tests of accuracy are reported next.

6.3.3 Judging the Adjudicator

Sed quis custodiet ipsos custodes?

(But who watches the watchmen?)

— Juvenal

Testing the Adjudicator presents a few complications, largely because style is difficult to define. Whereas the Examiner was designed to categorize its input into one of 26 categories built into it, the Adjudicator only "knows" one style category at a time — the one that represents the gridfont that Letter Spirit is handling in the current run. This raises the question of how the Adjudicator could be said to make a style categorization of any kind — right or wrong. One approach would be to use the score that the Adjudicator generates for an input, and to compare it to some fixed threshold, interpreting scores below that threshold (again, lower scores are better) as a statement by the Adjudicator that the input belongs in the style indicated by the Thematic Focus. Then these answers could be compared with some person's notion of what the correct answers should be, and the Adjudicator's performance could be expressed by the proportion of its answers that are correct (in the sense of agreeing with the human judge).

A related, but better, approach is to "train" the Thematic Focus with the style of a few letters in a given gridfont, and then to have it rate several gridletters from the same letter category (different from any in the training set). One of the tested gridletters would be from the same gridfont as the training set. Whichever gridletter the Adjudicator gave the best rating to would be its answer in a multiple-choice test, of sorts. In essence, the designer of the gridfont is utilized as the authority on which of the test gridletters best fits the training set's style.

The most comprehensive possible test of this kind would try all combinations of all possible sets of letter categories for training, all possible sets of letter categories for testing, and all possible sets of fonts upon which performance is tested. The number of permutations that this would involve is intractably large, however, and so the test that was actually carried out used 21 test sets that captured a wide variety of possible permutations. Each test set used five gridfonts, four *training* letter categories and, where possible, four *testing* letter categories. For six of the test sets, the set of five gridfonts did not have eight total letter categories that the Examiner could reliably recognize from each of the five fonts. Any shortfall led to a reduction in the number of letter categories tested (from four down to two or three).

6.3.4 Style categorization on the example gridfonts

The entire test amounted to 370 multiple-choice questions, with five choices per question, put to the Adjudicator. The raw result is that the Adjudicator chose the "correct" answer 60.8% of the time. The results by test set appear in Table 6.2. The gridfonts tested are 21 of the 23 introduced in Chapter 1. (Examiner performance was so poor for the remaining two fonts — Sluice and Three D — that they cannot be used in tests of the Adjudicator or of any part of Letter Spirit that depends upon the Examiner.) In the table, the gridfont names are abbreviated.

6.3.5 Understanding Adjudicator performance

The raw score of 60.8%, while far better than the 20% that would result from picking by chance, is not spectacular. However, there are mitigating factors. In general, style is less well-defined than letter category, and it has already been discussed how difficult it is to give Letter Spirit a strong sense of letter category. The most extreme demonstration of the blurriness of style is that there are plentiful examples where a

Set	*Gridfonts*	*Train*	*Test*	*Score*
1	BL BR BT CL HO	bcnp	agoz	100.0
2	HU4 SH SLT SN STSQ	dhrz	amsu	75.0
3	HI4 CH DBS FLR FNT	dfqt	vy	50.0
4	BW BT SAB SLS SQC	flty	hijk	75.0
5	BL INT HO SH SN	bdfo	grxz	60.0
6	HU4 BR SLT CL STSQ	dgiq	amjz	55.0
7	BL WAR HU4 BW HI4	fily	juvw	30.0
8	BR WAR SH CH INT	cflr	ioqz	55.0
9	BT SLT DBS SAB HO	eghi	loqs	80.0
10	CL SN FLR SLS STSQ	bdgi	klmp	65.0
11	FNT SQC STSQ SH BL	bdft	hnrw	40.0
12	HU4 BR DBS WAR SLS	fghi	bcde	80.0
13	HO SN FLR SAB INT	jlqr	bcdf	30.0
14	BT WAR SLT FNT SQC	hnry	dfw	60.0
15	HI4 DBS SLS SLT CL	cdjq	elpx	80.0
16	BW HU4 SN INT SAB	jlty	fi	40.0
17	SQC WAR FNT CL CH	dfhn	ry	50.0
18	BR CH DBS BW BL	fhil	tvxy	65.0
19	CH BT FLR HI4 FNT	dfry	qtv	53.3
20	HO SAB HI4 SH SQC	dfil	bnps	35.0
21	BW INT FLR SLS STSQ	fijl	ty	90.0
All				60.8

Table 6.2: Adjudicator performance.

given letter category is filled by the same gridletter in each of two or more gridfonts. The same 'x' can be found in Close, Hint Four, Hunt Four, Shorts, Slash and Standard Square. As a result, in test set 15 alone, the program is asked six times to choose between 'x's that are completely identical. Other examples of the same phenomenon are scattered through the test.

A related effect, but a bit more subtle, occurs when the gridletters the Adjudicator must pick from are not identical, but similar. Test set 11 shows a large number of errors, despite the fact that some of the gridfonts in the test set are easily distinguished when tested in the context of different test sets. The reason for this is that three of the gridfonts in the test set are defined, in part, by squareness — by right angles, by the scarcity of diagonal quanta and of angles of 45° or 135°. The similarity of the three gridfonts (Standard Square, Square Curl, and Shorts) complicates the discrimination task considerably. By analogy, consider a hypothetical test in which a person is shown one of their friends at a modest distance and asked to identify the friend by name. The more the friends in question look alike, the more difficult the test becomes. Similarity between the objects to be discriminated among is the very problem that the Adjudicator faced with test set 11. For the three "squarish" gridfonts, the Adjudicator gave the correct answer in only 4 of 12 possible opportunities in test set 11 (33%), but 29 out of 43 opportunities in all other test sets (67%). This effect was most pronounced in the case of squarish styles, but it cropped up elsewhere as well.

An interesting fact about squarishness is that it is probably the easiest motif-based style for the Adjudicator to recognize, so long as the Adjudicator is not asked to discriminate between two squarish gridfonts. Squarishness has a greater degree of symmetry than other motifs, and so squarish variable motifs can be matched against a squarish gridletter in many ways, not just the few ways that occur with less symmetric motifs. Consequently, tests of squarish gridletters with styles based on squarish motifs produced unusually good scores.[4]

Table 6.3 shows the performance of the Examiner and the Adjudicator on each gridfont. Here, the Adjudicator scores indicate the percent correct when four gridletters from the given gridfont were used to *train* on. Then, the Adjudicator was used to score five gridletters from one single letter category — one of which belonged to the same gridfont as the training set, and is therefore the "correct" answer.

The best way to gauge the capabilities of the Adjudicator is to pinpoint which kinds of styles it is capable of recognizing accurately, and which it is not. Table 6.4 lists the tested gridfonts, with a brief assessment of the kind of style each employs, along with the Adjudicator's performance on trials for which that style was the training set.

[4]It was probably the inter-gridfont confusions among the three squarish styles that kept the Adjudicator from achieving a much higher performance on them.

	Examiner	*Adjudicator*
Benzene Left	95.4	90.0
Benzene Right	99.6	80.0
Boat	97.3	83.3
Bowtie	60.4	56.3
Checkmark	76.5	33.3
Close	94.6	77.8
Double Backslash	83.8	94.4
Flournoy Ranch	76.5	66.7
Funtnip	54.2	28.6
Hint Four	88.8	29.4
House	94.2	75.0
Hunt Four	95.8	27.8
Intersect	63.5	37.5
Sabretooth	81.9	33.3
Shorts	97.7	45.0
Slant	90.8	89.5
Slash	85.8	86.1
Snout	96.2	50.0
Square Curl	81.9	64.7
Standard Square	100	72.2
Weird Arrow	87.7	41.1
Mean	85.8	60.1

Table 6.3: Adjudicator performance on 21 gridfonts.

Performance is boiled down to success on a three-valued scale of "strong" (at or over 80%), "moderate" (between 60% and 80%), and "weak" (under 60%). For the style descriptions, "invariable motif" refers to a motif that does not change in location or orientation — what the program represents as a literal motif. "Variable motif" refers to all other kinds of motif. "Abstraction" does not refer to an abstract rule, but to a style that can be represented only as an abstraction of a higher order than those built into Letter Spirit.

The immediate result is that the Adjudicator performs very well only on gridfonts defined primarily by an invariable motif. Performance on these styles was 84%, while performance on all other styles was just over 50%. Of the styles *not* defined primarily by an invariable motif, the ones upon which the Adjudicator performed best were

	Style Type	*Adjudicator performance*
Benzene Left	invariable motif	strong
Benzene Right	invariable motif	strong
Boat	invariable motif	strong
Bowtie	variable motif	weak
Checkmark	variable motif	weak
Close	invariable motif, abstraction	moderate
Double Backslash	invariable motif, secondary motif	strong
Flournoy Ranch	abstraction	moderate
Funtnip	variable motif, abstraction	weak
Hint Four	weak motif, abstraction	weak
House	invariable motif	moderate
Hunt Four	weak motif, abstraction	weak
Intersect	abstraction	weak
Sabretooth	weak motif, abstraction	weak
Shorts	invariable motif, abstraction	weak
Slant	invariable motif	strong
Slash	invariable motif	strong
Snout	variable motif	weak
Square Curl	variable motif	moderate
Standard Square	invariable motif, abstraction	moderate
Weird Arrow	variable motif	weak

Table 6.4: Style vs. Adjudicator success.

those with an invariable motif as a secondary attribute (especially Close and Double Backslash).

The gridfonts without an invariable motif that led to the best Adjudicator performance were Square Curl and Flournoy Ranch. Perhaps, in the case of Square Curl, this is because it comes close to having a prevalent invariable motif, because so many of its gridletters use portions of the square o-ring motif that is simply the perimeter of the central zone, even if none of its gridletters but ‘o’ actually use that motif in its entirety. Also, performance on Square Curl is boosted by its staunch adherence to abstract rules suppressing diagonals and angles other than right angles. Flournoy Ranch is a somewhat unexpected success story. Its style emerges from a hodgepodge of norm violations and small variable motifs. It was probably a beneficiary of the idiosyncrasy of its style; just as the squarish fonts led to reduced performance because of the ease with which they could be confused with one another, Flournoy Ranch probably led to relatively good performance because its gridletters were hard to confuse with anything else. An interesting perspective is that Flournoy Ranch may also benefit from the fact that it is the combination of a number of substyles, no single one of which is indicative of the overall style. It was shown in earlier discussion using Shorts as an example that the Thematic Focus tends to represent styles as combinations of substyles even when a person might be able to describe the same style more succinctly. Flournoy Ranch, then, perhaps plays to the Adjudicator's strength by actually *being* a combination of diverse stylistic elements.

6.3.6 Stability of the Adjudicator

The nondeterministic nature of FARG models carries with it many benefits, but some hazards as well. If the behavior of the Adjudicator were excessively erratic, that could undermine performance of the larger Letter Spirit program. For example, if the Examiner and the Adjudicator happened by chance to give one of the Drafter's renderings of a gridletter a score that was better than they should give it, that might make the top-level program decide that the gridletter should be kept as a member of the gridfont that it is designing, to the exclusion of other gridletters that are actually better. The danger is that even if the Adjudicator (or Examiner) gave the worse gridletter a worse rating nine times out of ten, one fluky score could lead to an inferior choice becoming a permanent part of the gridfont. What follows are the results of a few tests that were conducted to check whether the Adjudicator's ratings are relatively stable across a variety of circumstances.

Run-to-run stability

One design factor in the Adjudicator is the number of SP-detecting codelets that are posted to the Coderack at the beginning of a run. With too small a number, important SPs would not be detected, but on the other hand, having too high a number would waste computational resources while adding no functionality; for example, it does not matter if the average SP that is present is detected two times or a hundred, but the added effort in the latter case suggests an egregious inefficiency.

If the number of SP-detecting codelets were too small, then the accuracy of the Adjudicator would vary considerably from run to run, with success coming mainly in those runs where the Adjudicator was lucky enough to find, despite its limited opportunities, the SPs that happened to be important for the given run. In order to test the stability of Adjudicator results, a test set was run with each gridletter being tested three times, and the best of the three scores was then compared to the score for the same test set based on a single run per gridletter. If the amount of exploration that the Adjudicator performs were short of what is adequate, then random noise would affect the single-run test more than the triple-run test.

This test was conducted on test sets 2, 4, 9, 10, and 12 from Table 6.2. The single-run test achieved 75% accuracy to the triple-run test's 73%. Clearly, this does not demonstrate any benefit to triple runs, and so it appears that the Adjudicator's results are fairly consistent from one run to another.

Training set size stability

Another test explored the effects of training-set size on Adjudicator score. Here, it seems obvious that smaller training sets must lead to worse performance. Table 6.5 shows Adjudicator recognition performance on a pair of test sets, with variation in training-set size. The clear result is that a training set of size 1 produces recognition no better than (and in fact, here, somewhat worse than) chance. Once the training-set size reaches 3, performance is about the same as for training sets of size 5. Tests in Chapter 8 gauge the performance of the full Letter Spirit program with training-set sizes ranging from 0 to 25. The test reported in Table 6.5 is the first indication that a Thematic Focus trained on just one gridletter is not enough for reliable performance. As a result, runs in which Letter Spirit tries to create an all-new style from scratch, without any input seed gridletters, require some luck to get past the early, difficult stage, in which style has not been adequately determined.

Test fonts	*Test letters*	*Train: b*	*Train: bce*	*Train: bcdefg*
CL SN FLR SLS STSQ HU4 BR DBS WAR SLS	jklmp jlpvy			
Total		14%	60%	62%

Table 6.5: Adjudicator performance with varying training set sizes.

6.4 Conclusion

The Adjudicator is a qualified success. In multiple-choice tests, it shows a decent ability to handle certain styles. Its errors often occur in the reasonable circumstance that the choices between which it has to decide are similar to one another.

A second source of errors is the limited set of stylistic properties built into the Conceptual Memory. Lacking elaborate ways to generate novel SPs in the midst of a run, as the situation might call for, the program has to make do with a fairly restricted set of SPs — certainly more limited than those that a person might make use of. A person has the ability to generate new styles and new ways of representing style (the immense complexity behind this was touched upon in Chapter 3); the Adjudicator can only generate new styles in limited ways — namely, through the limited means by which novel SPs can be produced (mainly motifs) and by representing styles as agglomerations of SPs (i.e., of substyles). As a result, the ways (mentioned in Chapter 1) that human-designed gridfonts can base a style not on the mere presence of an SP, but also on its context, salience, or manner of affecting the flow of a gridletter are all beyond the Adjudicator.

The third source of errors is caused by what is perhaps the most deeply-rooted problem in the Adjudicator. Although the Thematic Focus, with its organization into levels that distinguish the importance of SPs, shows some success in terms of Adjudicator performance, it fails to truly focus upon *the* most important SPs for any given style. This is due to many reasons. The Adjudicator lacks the ability to formulate arbitrary combinations of stylistic properties, and the set of primitive stylistic properties with which it has been imbued is certainly no match for the rich, dynamic palette that people possess. Although the Adjudicator can form a representation that combines SPs, it was shown (using the example of Shorts) that this is not always an adequate substitute for representing the style correctly.

Many of the Adjudicator's weaknesses may stem from its lack of human abilities that are based upon algorithmic mechanisms. This is illustrated in the fact that the Adjudicator lacks the ability to build structures representing new SPs out of old ones and by modifying them with properties like context, salience, and flow. This kind of structure-building, a part of FARG projects like Copycat, could be a powerful addition to Letter Spirit in future work. Meanwhile, the six-level Thematic Focus is perhaps too "mushy" a way to represent the difference, in any given style, between the most important SPs and relatively unimportant ones. Symbolic AI programs often suffer from brittleness — a rigidity whereby they are too dependent upon precise conditions. The Thematic Focus perhaps goes too far the other way, by placing importance on a continuum where the top is never adequately distinguished from the middle. In the spirit of the dual mechanisms introduced in Chapter 2, an interesting solution might be to combine the current "mushy" Thematic Focus with an algorithmic approach that might be too brittle to represent style well on its own, but that might cooperate well with a Thematic Focus like the current one.

Despite its shortcomings, the Adjudicator nonetheless manages a respectable performance in representing style and categorizing on the basis of style. It is important to remember that another vital purpose of the Adjudicator — the construction of a representation of style for the purposes of *creating* new gridletters — has not been tested in this chapter in any way. An outstanding style categorizer that did not build a representation of style for use by the Drafter would be of little use to the Letter Spirit project. Because the Thematic Focus is the same representation used for style categorization and for incorporating a goal style into new gridletters, one might suspect that the same deficits listed above must manifest themselves in the creative task as well. Surprisingly, as the next two chapters will reveal, this is not always true.

6.4.1 The module as a model of human behavior

The strictly modular structure of Letter Spirit is probably a weakness in terms of making the Adjudicator an excellent model of human behavior. A person faced with the Letter Spirit challenge is not obliged to view all the seeds first, form an impression of style based on them, and then never look at them for the rest of the gridfont-designing task. That is, however, largely what the Adjudicator does; each individual gridletter accepted as a member of the style has only one opportunity to influence the program's representation of style. Perhaps people do work with style in this way at times, but they can also operate in other ways, such as re-representing the style of the seeds later during the design of a gridfont, or making comparisons between pairs of gridletters, temporarily ignoring the other ones that are available. The Adjudicator never returns to previously-accepted gridletters to reevaluate how they should influence the program's representation of style. (The Drafter uses information in the Thematic Focus and the Library to create new gridletters, which may *indirectly* lead to further evolution of the style, but a gridletter is never reevaluated so as to *directly* alter the contents of the Thematic Focus or the Library a second time.) Further work on Letter Spirit must focus on tying together the various subtasks in gridfont design in ways that are more complex than the current modular architecture can handle.

In the larger framework of aesthetic perception, the Adjudicator shows the ability to recognize visual coherence. Future work might investigate the aesthetics of visual style, identifying how different styles have different feelings attached to them — which are serious, which are playful, and so on. This would benefit from an increased understanding of how, in general, emotion is involved in cognition — something that is poorly understood at this time.

CHAPTER SEVEN

Drafter

7.1 Introduction

All of the work done by Letter Spirit in actually rendering a gridfont is carried out by the Drafter. On the surface, one might even say that with the implementation of the Drafter, the Letter Spirit challenge has been met, because one can obtain gridfonts simply by running the Drafter on each letter category; and, if the Thematic Focus already contains style information for some seeds, then the Drafter's work will even have a measure of coherence. The quality of the Drafter's work is, however, dramatically inferior to that of Letter Spirit as a whole. The magnitude of the difference between them will be considered in the next chapter; for now, it should suffice merely to warn the "squeamish" that, although the output seen in this chapter takes the form of gridletters (or rather, attempts at gridletters), many of those shapes will be highly unacceptable as members of their intended letter and style categories.

The architecture of the Drafter differs drastically from that of the other two modules, because the Drafter can work on any of three different levels, and only one of the levels involves a Coderack-based architecture. Like the Adjudicator, a single run of the Drafter may be relatively simple, but tremendous complexity comes out of multiple runs.

7.2 Implementation

7.2.1 An overview of processing in the Drafter

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

1. Drafting a gridletter

2. Drafting a role-filler

3. Drafting a single quantum

For the Drafter to render a gridletter, it must act on the top level. There are two ways to go about this. One is to create the gridletter in a single step, by borrowing and adapting a letterform from the Library. The alternative to that is to create the gridletter one role-filler at a time. At that level, a similar set of choices is repeated. The Drafter may borrow a role-filler wholesale from the Library, or it may create the role-filler quantum by quantum.

The drafting of a quantum is carried out by a run of the Coderack, which selects the next quantum for the role-filler out of all the eligible candidates. In this mode of drafting, role-fillers are drawn one quantum at a time, via successive point-to-point moves, as though an ink pen were doing the drawing. A run of the Drafter Coderack begins with the pen in some location, with various amounts of information on where and why it should draw next, and the task of one run of the Coderack is to decide which quantum, if any, should be drafted next.

For the purposes of the rest of this thesis, these three types of activity are defined as the modes of the Drafter. Future development of Letter Spirit may warrant a different way of describing the program's organization.

7.2.2 Choice of methods

When the Drafter is given a letter category to attempt to draft, it chooses, nondeterministically, which mode or modes of Drafting to employ. First, the decision is made whether or not to create the entire gridletter in one swoop by whole-gridletter borrowing — that is, by borrowing and modifying a gridletter that has already been accepted into the current gridfont, but for another category. If this is not possible (there may not be any appropriate letter category to borrow from), then role-by-role methods must be used. If whole-gridletter borrowing is possible, there is a 75% chance that the gridletter will be rendered by whole-gridletter borrowing, and a 25% chance that role-by-role methods will be applied, anyway.

When a role is to be drafted, if it cannot be borrowed from the Library, then it must be drafted by Coderack drafting. If it can be borrowed from the Library, then there is an 85% chance that it will be rendered by borrowing, and a 15% chance that Coderack drafting will be used.

Earlier writings on Letter Spirit describe a fourth module, called the *Imaginer* [Hofstadter and FARG 1995]. The Imaginer would have made high-level decisions regarding the design of the gridletter, making recommendations on the letter and role-filler levels, without considering the specifics of the grid. The recommendations of the Imaginer would then be handed to the Drafter, which would try to implement them on the grid. In the current architecture, the decisions regarding whether to undertake borrowing or Coderack drafting may be thought of as an embryonic Imaginer. Subsequent work on Letter Spirit may develop a full-fledged Imaginer, and take these functions away from the Drafter.

7.2.3 Whole-gridletter borrowing

The Drafter can exploit various shortcuts to drafting that circumvent the usual Coderack-based drafting of the FARG-style module. Whole gridletters, for instance, can be borrowed from the Library and manipulated in ways that are hard-wired into the program. Thus, an attempt at 'd' may be drafted by simply taking the mirror-reflection of a 'b' that is already in the Library. There are 37 such heuristics built into Letter Spirit, shown in Table 7.1. When a method of borrowing is symmetric (flipping and rotating by 180°), it is allowed in both directions, although only one is listed in the table.

Goal	*Source*	*Operation*
a	e	rotate 180°
b	d	flip horizontally
b	p	flip vertically
b	q	rotate 180°
c	o	remove right-edge
d	p	rotate 180°
d	q	flip vertically
h	b	remove baseline quanta
h	y	rotate 180°
i	j	remove descender quanta
m	w	rotate 180°
n	h	remove ascender quanta
n	o	remove baseline quanta
n	u	rotate 180°
o	b, d	remove ascender quanta
o	g, p, q	remove descender quanta
p	q	flip horizontally
r	c	remove baseline quanta
r	n	remove rightedge
s	z	flip horizontally
u	o	remove x-height quanta
u	y	remove descender quanta
y	g	remove x-height quanta

Table 7.1: The ways in which the Drafter can attempt a new gridletter based on an old one.

In the current implementation of Letter Spirit, employing the exact same borrowing technique twice in one gridfont will always be redundant, producing the exact same new candidate gridletter. Therefore, once a specific borrowing technique has been utilized, it is removed from consideration for the rest of the run. Future changes to the top-level control may warrant a different policy regarding repeated whole-gridletter borrowing.

Drafting by means of borrowing and altering a whole gridletter is not guaranteed to produce gridletters of high quality. In full-fledged typeface design, it is even less likely to do so, due to subtle asymmetries in serifs, thicknesses, and the way strokes terminate [Jaspert, et al. 1986]; in the restricted world of gridfonts, these matters are non-issues. On the grid, the practice is in fact quite helpful; for example, in the 23 examples of human-designed gridfonts, 'b' and 'd' are mirror images of each other 16 times. Although whole-gridletter borrowing is not implemented in a way that is interesting as a cognitive model, it nonetheless faithfully reproduces an important technique in gridfont design as carried out by humans.

7.2.4 Role-filler borrowing

The borrowing of individual role-fillers is very similar to that of whole gridletters, except that a role-filler is borrowed as is, with no alterations. (Future work could explore richer kinds of role-filler borrowing such as creating a left-post by taking the mirror reflection of a right-post in the Library.) Role-filler borrowing may select any entry in the Library from a list of roles deemed "borrowable" for the desired role category. For any given role, this is a short list of roles, including the role itself, plus any highly related roles that might exist. For example, the cap of an 's' and a z-cap may be identical in shape within the same gridfont; this is true in 7 of the 23 sample gridfonts. This type of borrowing across role-categories is especially apt when two roles differ primarily only in terms of neighborhoods and contact, which do not directly affect the shape of the role-fillers. Table 7.2 lists all the role-filler borrowing allowed by the Drafter, by roles.

Objective Role	*Sources*
a-arch	a-arch, right-buttress
backslash	backslash
cap	cap, z-cap, crossbar
circle	circle
crossbar	crossbar, cap, z-cap
dot	dot
foreslash	foreslash
halfpost	right-halfarc, left-halfarch, right-halfarch, right-halfpost
left-bowl	left-bowl, e-bowl
left-halfpost	left-halfpost
left-post	left-post
left-tail	left-tail
left-uparc	left-uparc
right-bowl	right-bowl
right-buttress	a-arch, right-buttress
right-curl	right-curl, right-hook
right-halfpost	right-halfpost
right-hook	right-hook, right-curl
right-post	right-post
right-tail	right-tail
z-cap	cap, crossbar

Table 7.2: The ways in which the Drafter can borrow role-fillers.

There have been other models of alphabetic design that undertake only something akin to this step. The extent to which these programs — Daffodil and *abcdefg* — are related to Letter Spirit is addressed in Chapter 8.

7.2.5 Implementation of Coderack drafting

7.2.6 Basic strategy

Think globally. Act locally.

— slogan for environmentalists, and for the Drafter

A role-filler can be rendered via multiple runs of the Drafter Coderack — roughly one Coderack run for each quantum in the role-filler. Each run of the Drafter Coderack works by first rating those candidates that may serve as the next step in drafting the role-filler (either any of several possible quanta, or else the decision to draft no more quanta at all) and then probabilistically choosing among the rated alternatives. The Drafter Coderack is initialized and run many times per gridletter; to create a Standard Square 'b', the Drafter Coderack would run twelve times (one for each of the ten quanta, plus two that would make the decision to stop drafting when both roles seemed adequately filled).

7.2.7 Memory structures in the Drafter

Workspace

The Workspace is a copy of the grid, upon which the Drafter attempts to render a new gridletter. Several variables hold the information needed by the various modes of the Drafter to keep track of the state of the drafting process.

A key concept for Coderack drafting, which drafts from point to point on the grid, is the *current-point*, a point on the grid from which the next step will be taken. Also vital for Coderack drafting is the *candidate list*, a list of the candidates for the next step that the Drafter will choose. Every quantum that leads away from the current-point (without doubling back upon portions of the role-filler that have already been drafted) is a candidate. For each candidate, the candidate list has two numerical scores that estimate how appropriate the candidate is as the next step to be taken. The first, the *quit-sugar* score ("sugar" is meant to indicate that this is a measure of the desirability of the candidate), rates how well the candidate would serve as a step that would complete the drafting of the role-filler; the second, the *continue-sugar* score, rates the candidate as a step that would be part of the role-filler, but would not make it complete. The exact way in which the candidate list is computed and then used is described shortly.

A few other variables are needed for the Drafter to undertake the complex process of coordinating what potentially involves many different steps on different levels. For instance, one gridletter might be drafted by the borrowing of one role-filler and the Coderack drafting of two others, with each of those operations involving multiple runs of the Coderack.

- *the-whole*: the role-set that is being drafted
- *the-role*: the role that is being drafted
- *roles-drafted*: the roles (if any) from the-whole that have already been drafted
- *candidate list*: the list of possible choices for the Drafter's next quantum-level step, along with a pair of sugar scores for each
- *touch-points*: points that previously-drafted parts' norms say should be in contact with other role-fillers
- *touch-quanta*: quanta that previously-drafted parts' norms say should be in contact with other role-fillers

- *avoid-points*: points that previously-drafted parts' norms say should not be in contact with other role-fillers
- *avoid-quanta*: quanta that previously-drafted parts' norms say should not be in contact with other role-fillers
- *own-stuff*: the quanta that have already been added to the role-filler that is currently being drafted (if any)
- *other-stuff*: the quanta that make up the role-fillers (if any) previously drafted for this gridletter
- *current-point*: the point from which Coderack drafting proceeds
- *last-point*: the previous current-point
- *norm-curve*: the curvature that is the norm for the role being drafted
- *quit-points*: a set of points on the grid, weighted to express how well each could serve as the point at which drafting of the current role-filler could terminate
- *quit-flag*: true if the previous step was tentatively chosen as the final quantum to be added to the role-filler
- *quit-score*: if quit-flag is true, this expresses how strong the preference is that the previous quantum be the final one added to the role-filler

Not all of these variables are used by all levels of drafting. Whole-gridletter borrowing only uses *the-whole*. Role-filler borrowing only uses *the-whole* plus *the-role* and *roles-drafted*, although it also alters the content of the touch- and avoid- lists. Coderack drafting accesses all sixteen of the variables in the list.

Two of the aforementioned variables merit particular attention — current-point and quit-points. Before Coderack drafting can begin, the Drafter must determine where the drafting should begin, and a list of tentative locations where it may end. Each point on the grid is given a rating as the starting point and, separately, as a finishing point. These ratings are calculated by assessing the prominence the point has in the role's norms for tips and ends as well as in the quanta mentioned in motifs and abstract rules in the Thematic Focus. Then, the starting point for the role-filler is determined by spinning a virtual roulette wheel that chooses among all points, with each point's chances of winning being based on its rating as the starting point; the winner of this biased-random selection is the first current-point. The quit-points are the set of points that have nonzero ratings as the finishing point; a nondeterministic selection procedure picks one out as the point towards which drafting generally "aims" as it drafts its way across the grid from the starting point. It is not necessarily the case that this is where the drafting of the role-filler will actually end, although it often turns out that way.

Coderack

The Coderack mode of the Drafter does not operate in a complex way. It begins with a set of codelets that are run one at a time, with few if any new codelets being posted during the run. Its sole purpose is to calculate a set of scores (namely, a pair of sugar scores for each candidate). It begins fully loaded, with several of each type of Drafter codelet, and it runs those one at a time until they are all gone; a predictably small number of codelets may be added along the way, but flow of control in the Drafter basically proceeds as simply as water draining out of a bathtub.

7.2.8 Codelets

Drafter codelets are much more homogeneous that those of the other Letter Spirit modules, or in other FARG models. Each has the same general purpose — it has a small say in rating the attractiveness of all the candidates for the next single-quantum step in drafting. A codelet adds or subtracts to the quantity of *sugar* for each candidate in the candidate list. A run of the Drafter Coderack performs many such acts of addition, leaving score totals that assist the eventual choice among the options, with the selection being weighted by those scores.

One new term to be introduced here is "trait". A trait is much like a norm, but it is a norm modified by a norm violation. A norm is what is expected of a role when there is no additional information regarding style. When a style with a norm violation is specified, then something other than the norm — namely, the norm as *modified* by the norm violation — should be expected. In order to combine the letter definitions in the Conceptual Memory with the style definition in the Thematic Focus, the Drafter uses traits, rather than norms, as a guideline for design. Two codelet types utilize traits.

The function of each Drafter codelet type is listed below; the homogeneity of the set is obvious.

curve-following codelet (no arguments)

This codelet adds points to both sugar scores of each candidate based on how well it would make the role-filler that is being drafted follow the role's curvature norm on a path from its initial point to its tentative finish point.

inertia codelet (no arguments)

This codelet adds points to both sugar scores of each candidate based on how well it leads in the same direction as the previous quantum added to the role-filler that is being drafted. The magnitudes of the points given out by this codelet are relatively small, so that it acts more as a tie-breaker (favoring inertia over change in direction) between otherwise roughly equal choices.

motif-rewarder codelet (arguments: motif, level of motif in Thematic Focus)

This codelet adds points to both sugar scores of each candidate, based on how well the candidate would make the role-filler that is being drafted match the given motif. Motifs that are higher in the Thematic Focus are deemed to be more important, and therefore lead to higher point totals.

rule-enforcer codelet (no arguments)

This codelet chooses an abstract rule probabilistically from the Thematic Focus (favoring those in higher levels), then adds points to both sugar scores of each candidate based on how well it would make the role-filler that is being drafted obey the abstract rule.

trait-progress-reward codelet (argument: trait dimension)

It will already have been decided which trait is most preferable for the role-filler in terms of the given trait dimension — in the absence of any pertinent norm violations in the style, this will simply be the role's norm in that trait dimension; otherwise, it will be the norm modified by a norm violation. This codelet then adds points to the continue-sugar score of each candidate based on whether or not the role-filler that is being drafted can possibly match that trait after the candidate (and other, hypothetical, subsequent drafting) is added to it.

trait-met-reward codelet (no arguments)

It will already have been decided which trait is most preferable for the role-filler in terms of the given trait dimension — in the absence of any pertinent norm violations in the style, this will simply be the role's norm in that trait dimension; otherwise, it will be the norm modified by a norm violation. This codelet then adds points to the quit-sugar score of each candidate based on whether or not the role-filler that is being drafted matches that trait with the addition of the candidate (without supposing any possible future drafting).

finisher codelet (no arguments)

This codelet adds points to the quit-sugar score of each candidate based on how well its endpoint has been rated, before drafting began, as the finish point of the role-filler.

contact codelet (no arguments)

This codelet adds points to the sugar scores of each candidate based on how well the norms for role-fillers touching each other would be met by the addition of it to the role-filler that is being drafted.

quit-contact codelet (no arguments)

This performs the same task as the contact codelet, but expects the touching norms to be met exactly, and punishes very severely any candidate that does not do this.

As a further explanation, Table 7.3 summarizes on what basis each codelet type determines how much sugar it will add to (or subtract from) the candidates. Parentheses indicate that the codelet type subtracts, rather than adds, sugar.

Codelet type	*Rewards/(Punishes)*
curve-following	quanta on the path curving between the tips
inertia	(quanta that change the direction of drafting)
motif	quanta that make the part match a motif
rules	(quanta that make the part violate a rule)
trait-progress	quanta that do not make the part meet a trait
trait-met	quanta that make the part meet a trait
finisher	quanta leading to the finish point
contact	quanta that help incomplete part meet contact norms
quit-contact	quanta that help complete part meet contact norms

Table 7.3: The types of sugar in the Drafter and which quanta they give points to (or take from).

Those nine types of sugar are used to calculate the two sugar scores per quantum upon which the Drafter bases its choice of what step to take next. Each of the two sugar scores is the sum of a few of the nine types. Specifically, the quit-sugar score is the sum of seven sugar types: curve-following, inertia, motifs, rules, trait-met, finisher, and quit-contact. The continue-sugar score is the sum of six sugar types (curve-following, inertia, motif, rules, trait-progress, contact) minus 0.025 times finisher sugar type. (The last measure is to discourage non-final quanta from heading towards points that should ideally be reserved for the final quantum of the part.)

A Drafter run ends when the Coderack is empty, at which point one candidate is chosen. In most cases, that leads either to one quantum being added to the role-filler in progress, or to the decision that the role-filler is complete without the addition of any more quanta. The final choice of a candidate is probabilistic, carried out by the spinning of a virtual roulette wheel that is weighted by the sugar scores that were accumulated throughout the Coderack run.

Like the Adjudicator, the Drafter does not show great complexity in the way a single run takes place. The complexity of the work done by these modules accumulates over the course of multiple runs. In a full Letter Spirit run, the Drafter's Coderack might run as many as a couple of thousand times.

Other memory structures

The Drafter makes use of the definitions of roles and role-sets in the Conceptual Memory. These are precisely the same definitions used by the other modules.

The Drafter also accesses the Thematic Focus, which is built up by the Adjudicator. The Drafter uses the level of the Thematic Focus on which a stylistic property is found to determine how much influence that SP should have upon drafting. (Level information was also used by the Adjudicator to determine how much influence a stylistic property has — specifically, in calculations of temperature.)

Temperature itself is not a factor in the Drafter. All codelets have equal priority, which makes temperature irrelevant to the selection of codelets from the Coderack. Nor is temperature used as a scoring mechanism, as it is in the Adjudicator. In fact, the Drafter does not produce any kind of score in order to rate its own output; its output is rated only by the Examiner and the Adjudicator.

7.2.9 Topology types and Coderack drafting

It was mentioned in Chapter 5 that roles have one of four topology labels. The difference in the ways the four types of topology are handled is more pronounced in Coderack drafting by the Drafter than in the other two modules.

Segment is the most common type of topology, and is adequate for describing any role that consists of a single point-to-point stroke with, as a norm, relatively uniform curvature (whether it be straight, modest curvature, or pronounced curvature). 35 of 43 roles in Letter Spirit are of this type. The property of contact is used to discourage (but not prohibit) the Drafter from bringing a segment back around to touch or cross itself.

Slightly more complicated are *bisegments* — roles that are normally curved in one way through one portion of their length and in another way through the rest of their length. Examples of bisegments are f-post and right-curl — the stroke that makes up most of 'j'. Usually, the portion near one end of a bisegment has straightness as a curvature norm, while the rest of it is curved to one side. The four bisegment roles are those involved in 'f', 'g', 'j', 't', and 'y'. Bisegments, like segments, have norms not only for the locations and orientations of the two endpoints, but also for a midpoint. When a bisegment role is being drafted, the switchover from one norm for curvature to the other is made when drafting goes through or near the midpoint. Bisegments are treated exactly like segments with regard to the property of contact.

In a cruder implementation, bisegments might be treated as segments with just one curvature value for the entire length of the role. This might produce respectable results, but the fact that f-posts are more curved at the top than throughout the rest of their body seems worth incorporating into the model. Another possible approach would replace each bisegment role with two segments. For example, the f-post could be split into two roles — one for the straight portion and a curved one at the top of that. Aside from making the Examiner's segmentation be a bit more complicated to do correctly in this scheme, having each such stroke represented as two segment roles would not really be all that different from the bisegment approach.

Issues of curvature aside, segments and bisegments are both representations of point-to-point strokes. The remaining two types of topology norm are fundamentally different. A *loop* is a closed figure, such as the circle role used in 'o'. The other two loop roles are the down-circle of 'a' and the up-circle of 'e'. Where a segment or bisegment has norms for its two endpoints, a loop has norms for the points on what would normally be the leftmost and rightmost points on the curve. Loops, like bisegments, are drafted in two stages. The first stage is functionally like drafting a segment with strong-left curvature, from the leftmost point to the rightmost. Then, with the same curvature, the second stage drafts a segment from the rightmost back to the leftmost. Contact discourages a loop from touching itself during drafting, except that there is a very strong pressure for the drafting to end exactly where it started, so that the part becomes a closed figure, in the end.

The final topology type has only one representative in our alphabet, and the topology's name is the same as that of the role — *dot*. Dots, found on 'i' and 'j', are unique in that they may be point-to-point, closed, or may self-cross weirdly. The only real norms for them are their location (in the ascender zone) and size (usually, but not always, very small). Drafting a dot is handled in a point-to-point fashion, but the norms for dot endpoints do little to distinguish any of the points in the ascender zone from any other. Contact has no serious role as a constraint here, because human-designed dots show great variety with regard to self-crossing, closure, etc.

7.2.10 Parallel terraced scan

As was noted earlier, the flow of control in the Drafter is extremely simple; nonetheless, it does utilize the parallel terraced scan in one way. The list of candidates for which sugar is being calculated can be pruned, with any candidate whose total sugar falls far enough (4.0 points; example scores provided later will make the scale of that amount meaningful to the reader) behind the leaders getting removed. This will virtually never affect the decision made by the Drafter, because any candidate that is pruned would have had almost no chance of being chosen, anyway. This aspect of the program was motivated more by a desire to make the program run faster, rather than an attempt to increase cognitive plausibility; it simply takes less time to compute how much sugar each candidate should receive when there are fewer candidates. This is an application of the parallel terraced scan, in that each candidate receives attention proportional to the extent that it seems to be worthy of attention.

7.3 Performance

7.3.1 Step-level performance

The last three steps made by Coderack runs within a Drafter run are presented here in considerable detail. Figure 7.1 shows the state of the gridletter in progress before each of the three steps. The last step results in the decision not to draw any additional quanta, so the rightmost portion of the figure shows the gridletter in its final state. For this run, the program was trained on the style of the BCEFG training set for Benzene Right. The right-post of a ‘d’ has already been rendered (once it is finished, it makes no difference to subsequent activity whether it had been borrowed or whether it had been created by Coderack drafting), along with three quanta of the left-bowl. The three steps that finish the left-bowl are explained in detail, and this demonstrates much of the variety that can take place in Coderack drafting.

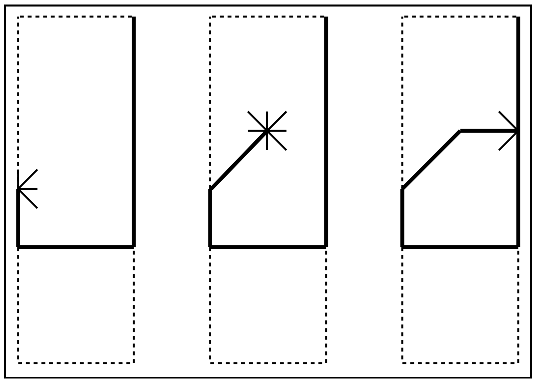

Figure 7.1: Three steps in Coderack drafting.

A reminder of the way quanta are numbered in the grid appears in Figure 7.2. The quanta that are candidates in these examples are shown, along with their numbers; it is by means of their numbers that they are referred to in the following tables and discussion.

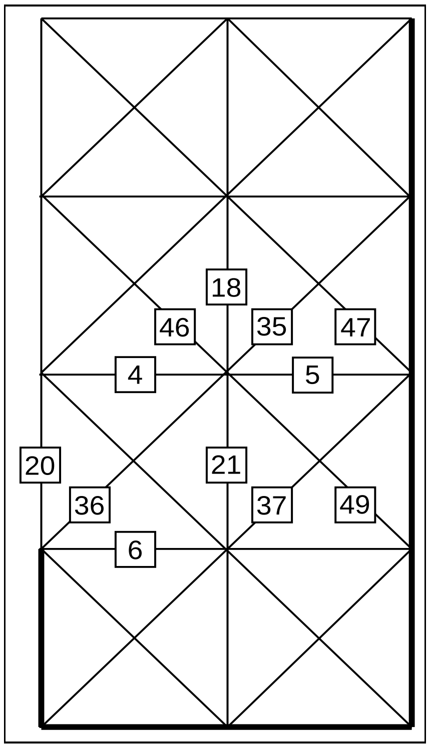

Figure 7.2: Key to the Drafter-steps example.

In the first example step, the current point is on the left side of the grid. The edge of the grid restricts the number of directions that this step can go, as do the quanta already present in the gridletter. For this reason, only three candidate quanta remain. Table 7.4 shows, for each candidate, the greater of its two sugar values at 30-codelet intervals throughout the Coderack run. In this example, the continue-sugar score happened to be highest in all instances. This should not be surprising, given that the role-filler cannot be completed very well with the mere addition of one quantum. In the race for highest score, quantum *36* gets out to an early lead, and maintains that lead throughout the run, finishing with a slight lead (only about 1% more sugar) over quantum *20*. Quantum *6* was farther back, and nearly fell behind by the 4.0 points that would cause it to be pulled from the race before the finish.

	30	*60*	*90*	*120*	*150*	*180*	*195*
20	14.3	19.4	29.7	44.7	59.5	74.4	74.5
6	13.8	18.7	28.6	43.1	57.5	71.8	71.9
36	15.1	20.5	31.0	46.0	60.6	75.2	75.3

Table 7.4: The sugar placed on each candidate through a Drafter Coderack run.

It is possible to scrutinize this activity even more closely. The raw scores do not make it clear *why* the quanta finished the race in the order that they did. Table 7.5 shows the breakdown of sugar scores, by sugar type and by candidate, as of the end of 30 codelets. (The constraint of page width forces a change from the organization from the other tables in this section, so the quanta list runs across, rather than down.) Curve-following (the measure of how well the candidate leads from the beginning of the role-filler to where it should perhaps end, while maintaining the proper trait for curvature), it can be seen, contributes most of the total sugar. The variation between the candidates' curve-following scores, in this case, is not nearly so large as the values themselves. Most other values are constant across the candidates. One glaring exception is trait-met, where quantum *6* receives a stinging rebuke! However, this is not part of the continue-sugar score; it is only part of the quit-sugar score, which is a statement that quantum *6* is not an appropriate way to terminate the role-filler. The only type of sugar, besides curve-following, that distinguishes any of the candidates from the others in terms of continue-sugar score is the motif sugar type, which rewards *36* with a 0.9-point advantage over both of its competitors. This is because the style in question is Benzene Right, and quantum *36* is part of the o-ring motif for that style. In fact, it is a part of all five seeds used to train the Thematic Focus in this example. Thus, between curve-following and motifs, Quantum *36* has a whopping 1.7-point lead on its competition after 30 codelets. This margin is repeated, the lead is extended, and quantum *36* wins the race on the basis of the fact that it fills out the desired curve nicely while also incorporating an important motif. Note that the edge that the winner got due to motifs was slightly larger than that due to curve-following, so that if the o-ring motif in question had been that of Standard Square, quantum *20* would likely have been the winner.

It should be pointed out that, while the selection of the actual winner on the basis of sugar scores is nondeterministic, it is weighted *heavily* by those scores, in a manner that is not linear but is weighted by sugar score to a rather large power (25). This means that the candidate with the highest score will usually end up being the candidate actually chosen, unless two candidates have scores that are unusually close.[1]

	20	*6*	*36*
curve-following	14.3	13.8	15.1
finisher	-0.1	-0.1	-0.1
inertia	0	0	0
trait-met	0.5	-384.0	0.5
trait-progress	0	0	0
rule-enforcer	0	0	0
motifs	0	0	0.9
contact	0	0	0
quit-contact	-0.2	-0.2	-0.2

Table 7.5: Candidates rated by sugar type.

The next step follows principles similar to those seen in the first example, but is a bit more complex. There are more candidates in this step, because the current point is now out in the middle of the grid and is thus less constrained. This time, many

[1] A 1% edge raised to the power of 25 becomes a 28% edge. If this transformation were not performed, a candidate with a 1% edge over a competitor would only have a 1% edge in being selected. Experiments determined that this did not give the sugar scores sufficient influence.

of the candidates will end up rated higher as final steps in the role-filler, rather than as steps in the middle of the role-filler. Those candidates are marked in Table 7.6 with an asterisk. In fact, all candidates that make the role-filler touch the right-post have high scores in terms of the quit-contact sugar type, and therefore it is exactly these candidates that are rated higher as final steps. Those candidates that do not lead to contact with the post are rated higher in terms of continue-sugar score. For the sake of brevity, the full breakdown by candidate and sugar type is not provided here. The essential result is that quantum *5* wins the race due to many factors, such as the following ones: it fulfills Benzene Right motifs, it has a good curve-following score, it brings the role-filler to a successful close by touching the right-post, and it touches it in a place where the finisher likes to see it finish (which is related to the norms for left-bowl tips). Quantum *5* essentially runs away with the prize, and as its lead grows, most of its competitors (as they fall 4.0 or more points behind) drop out of contention. Indeed, by the end, it is the only runner left in the race.

	30	*60*	*90*	*120*	*150*	*180*	*195*
*5**	5.5	11.1	23.1	24.0	43.8	59.5	78.9
*35**	5.1	10.4	20.0	20.6	40.5		
4	5.3	10.2	19.7	20.4			
21	5.3	10.2	19.6				
18	5.3	9.7					
*49**	5.1	10.5					
*46**	5.2	9.5					

Table 7.6: The sugar placed on each candidate through a Drafter Coderack run.

When a candidate wins on the basis of its quit-sugar score, the run does not truly end; experiments revealed that it was in fact best to try one more step. One might say that when a candidate has been chosen on the basis of quit-sugar, the rest of the run is on probation. The quit-flag is set to true, and the quit-score is set to the quit-sugar value that won the last round. In the subsequent step, a quantum is added to the role-filler only if its sugar score beats the quit-sugar value set previously. Otherwise, this step does not add a quantum to the role-filler, and the drafting of the role-filler is truly over.

The third Coderack drafting run of this example, then, is run under different conditions from the first two. Because the current point is in a location crowded both by the gridletter and by the edge of the grid, there are only two contestants. This time, it is not enough for a candidate to win the race in order to be included in the role-filler; the candidate must also beat the mark set by the previous winner. As Table 7.7 shows, quantum *37* wins its heat handily, knocking its only competitor out quite early on. Its sugar score of 72.0, however, does not reach the 78.9 needed to qualify it for the role-filler. At this point, the left-bowl is declared complete. Because both roles of the role-set, 'd1', are filled at this point, the gridletter is also complete.

	30	*60*	*90*	*120*	*150*	*180*	*195*
37	13.9	28.8	38.6	48.0	57.7	67.4	72.0
*47**	13.4						

Table 7.7: The sugar placed on each candidate through a Drafter Coderack run.

Of course, many issues that led to various details in the design of the Drafter did not come up in the previous example. An important one to mention is the issue of stability. In the discussion of the Adjudicator, it was seen that, given a fixed set of initial conditions, the Adjudicator's results were relatively similar, but not identical, from one run to another. The Drafter is less stable, and can depend greatly upon which stylistic properties are passed to the codelets as parameters. The outlandish score of -384 (in the context of the other scores) in the first example was due to a bit of a fluke in the selection of which traits to take note of. It does not mean that quantum *6* could never be chosen in similar repetitions of this situation, although it is unlikely, but in this run, it certainly is doomed to failure by the single −384 score. Achieving stability is the motivation for making the runs as long as they are. In the three examples presented here, the eventual winner was, in fact, always in the lead after a mere 30 codelets. One might see this and protest that the longer runs are a waste of time if the race is always decided so soon. The length of the runs is due to the fact that the race is *not* always decided so soon. Longer runs provide stability to guard against random flukes that might put an unusual candidate ahead too early.

7.3.2 Letter-level performance

Two ordinary styles

The previous example provided a zoom-in on the Drafter's quantum-by-quantum activity, with the end product of a complete gridletter. It is also important to show how a number of examples on the whole-gridletter level characterize the Drafter's aggregate behavior. Figure 7.3 shows a kind of "histogram" of Drafter output, displaying the most frequent results in 100 Drafter runs on each of three letter categories ('d', 'l', and 'm'). This demonstration of Drafter variety was derived entirely from Coderack drafting; borrowing inherently produces little variety, especially in the case of whole-gridletter drafting, which, relative to a given borrowing rule, returns the exact same new gridletter each time. The examples shown involve three different gridletters, each operating in the environment of a Thematic Focus trained on BCEFG for Benzene Right. Figure 7.4 shows exactly the same kind of output, but with Standard Square used as the training gridfont.

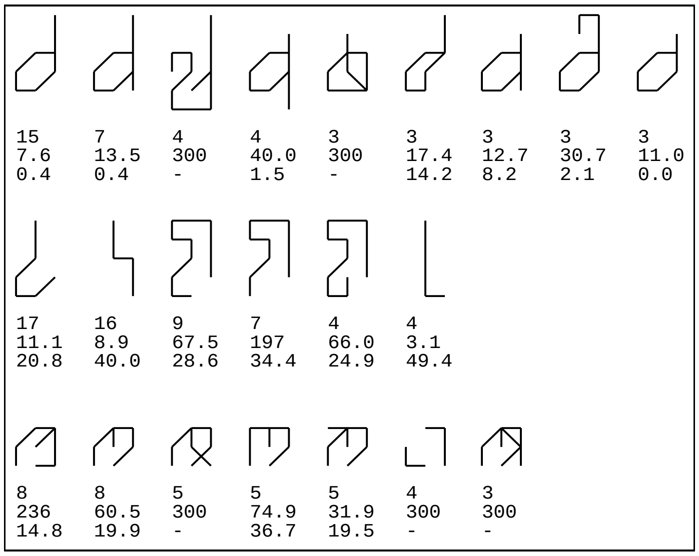

Figure 7.3: Drafter output for Benzene Right.

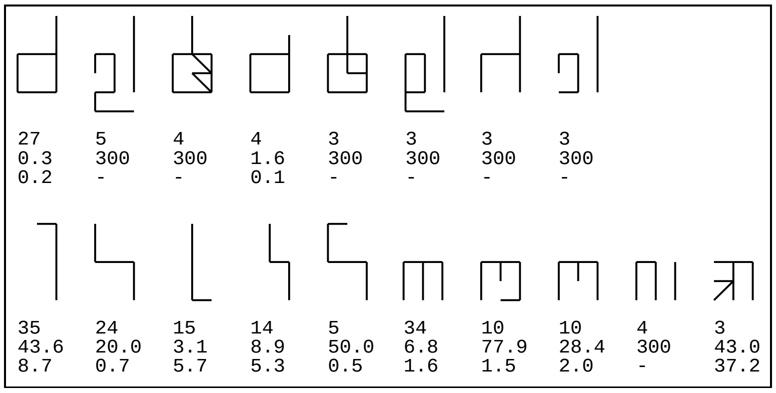

Figure 7.4: Drafter output for Standard Square.

These figures display each gridletter that got drafted three or more times. (Many other shapes came up once or twice, but showing all of these would make the figures overwhelmingly complex.) The first number below each gridletter is the number of times it was produced (again, out of 100 runs); the middle number gives the mean Examiner score for the gridletter; and the third number gives the mean Adjudicator score (omitted if the Examiner failed to recognize the gridletter as a member of the appropriate letter category). (As always, high scores from the Examiner and Adjudicator indicate a bad rating; low scores indicate a good rating.)

From these figures, a number of observations follow. One is that in each case, the Drafter produces considerable variety. When the Drafter makes several attempts at a category, it can be expected to produce several distinct outputs. Despite the variety, the cream seems to rise to the top in the sense that the best-rated shapes tend to occur quite frequently. In particular, in the test cases, the best-rated output in each situation occurred from 8% to 34% of the time. Consequently, the expected number of attempts that are necessary to generate a good attempt seems to be reasonably low. When the Drafter does generate a good attempt, the other modules are apt to recognize it — the best-rated gridletters in the test cases seem to a human eye to be good versions of their intended letter and style. Interestingly, though, the best-rated output is not necessarily that which is produced the most often.

In most of these ways (the point about other modules' evaluation does not apply), Drafter "histograms" are very similar to histograms of the output of FARG architectures such as Copycat [Mitchell 1993; Hofstadter and FARG 1995]. Copycat and the Drafter both build high-level structures that emerge from many low-level events that are purposeful but are guided by nondeterministic factors. These architectural commonalities lead to commonalities in the character of their output.

No style

Important to the next chapter, the Drafter does not require that the Thematic Focus have any contents in order for it to work. In such a situation, only Coderack drafting can take place, because an empty Thematic Focus entails an empty Library, so borrowing cannot be an option. Coderack drafting, by means of motif-rewarder, trait-progress-reward, trait-met-reward, and rule-enforcer codelets, normally accesses stylistic properties from the Thematic Focus, and sugar is sprinkled on candidates based on this, which is the way that Coderack drafting incorporates the trained style into its work. Drafting with an empty Thematic Focus simply occurs without those codelets adding any sugar; their net effect on the drafting is zero, so drafting is controlled only by the remaining factors: curve-following, tips, inertia, contact, and quit-contact. Figure 7.5 displays the result of trial runs identical to those shown in the two previous figures, but drafted with an empty Thematic Focus.

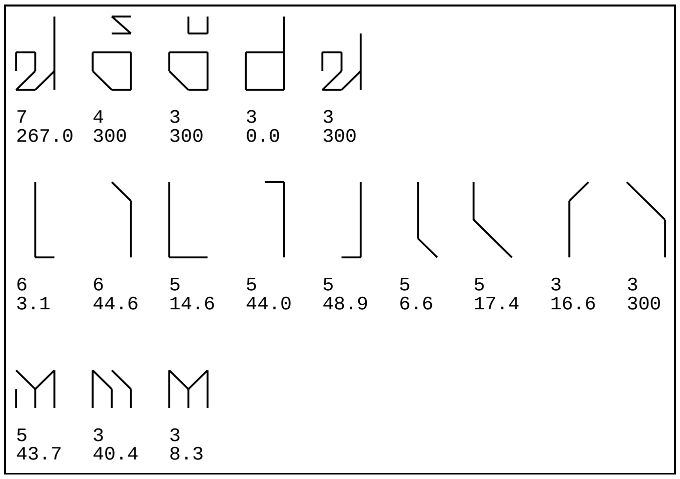

Figure 7.5: Drafter output for an empty Thematic Focus.

These runs are obviously less constrained than those with a non-null goal style, and this is reflected in the flatter distribution of results. There are no Adjudicator scores reported here, because with an empty Thematic Focus, all Adjudicator scores are the same (a bit over 50), to within the threshold of random noise. The next chapter shows the results of runs of the full Letter Spirit program where the program is not given any seed gridletters. In this circumstance, the first gridletter it produces in each run is a sort of self-seed, and Figure 7.5 shows the kind of self-seeds the program produces. Any shape that is rated well enough by the Examiner (again, the Adjudicator scores on such a run vary relatively little across gridletters and are therefore meaningless) might be used as the seed for a truly autonomous gridfont.

Chapter 8 presents Letter Spirit runs that begin without any human-designed seeds; for now, it suffices to see the variety that arises in unseeded drafting.

Style in the face of randomness

You don't have to be crazy to work here, but it helps.

— sign commonly seen in workplaces

As a final example, Figure 7.6 shows output for Benzene Right, but with the virtual knob controlling a variable involving the *randomness* of quantum selection turned up to a higher level. In a Letter Spirit run, this knob is gradually turned up throughout a run, to increase the chances that output produced late in the run is not mere repetition of output from runs performed already. The randomness level used to generate the output in this figure is the same as it would be after 200 gridletter-productions in a Letter Spirit run (with the maximum length of a run normally set at 300).

For 'd' and 'l', the shapes most commonly produced in the low-randomness run are also those most commonly produced here, but they occur only about half as often. For 'm', only one result came up as many as three times. In general, it is clear that the desired effect occurs — namely, that variety is increased. Chapter 8 includes a discussion on how turning up this knob throughout a run amounts to taking advantage of the parallel terraced scan, and thus improves overall Letter Spirit performance.

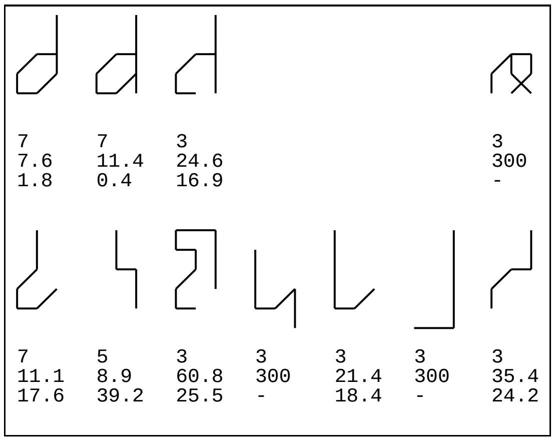

Figure 7.6: Drafter output with high randomization.

7.3.3 Gridfont-level performance

With the Drafter, the implementation of Letter Spirit could be called complete, if the desire were merely to have *some* kind of program that completes gridfonts based upon human-designed seeds. This section shows the kind of work such an approach leads to. It will be immediately apparent that the quality of the output is inconsistent at best, and the quality achieved by taking a single Drafter attempt as the program's output is quite poor. The full Letter Spirit program takes advantage of a strategy based upon review and revision, much as described in Chapter 4, and the superiority of this is made clear in Chapter 8.

Figure 7.7 shows five gridfonts based on BCEFG seeds from the SMALL set of gridfonts, with the remaining 21 letter categories filled in by the Drafter. A brief, comprehensive summary of this output is that it spans a wide range in terms of quality. Each gridfont is a mixture of letters that are good renditions of their intended letter category and style, and letters that are poor renditions in one or both of those aspects. (Of course, the seed categories are all filled with the human-designed input, which is uniformly good.) In the subjective estimation of the author, about a quarter of the

Drafter-designed gridletters are fine in terms of both letter and spirit, while about half fail in terms of letter, and the remaining quarter fail in terms of spirit. There *are* occasional flashes of high quality, such as the 'z's in Benzene Right, House, and Standard Square. Like most models of creativity mentioned in Table 4.4, the Drafter makes its choices in an informed way, and so it achieves some success. However, it is probably worse than most of those models in terms of how often its output is downright poor. The rest of this chapter explains what leads the Drafter to make such frequent blunders, and why, despite first appearances, this is not such a bad thing.

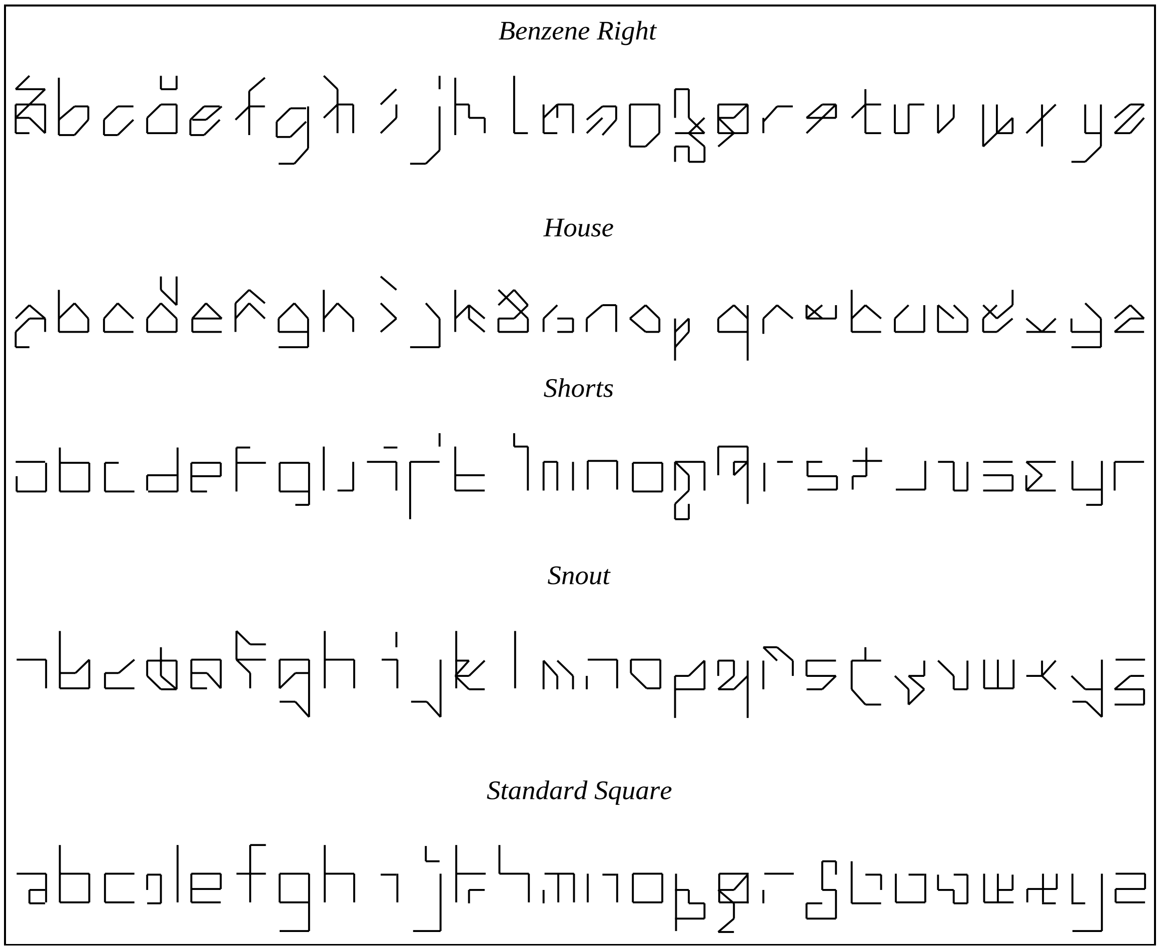

Figure 7.7: Drafter production of five gridfonts.

7.3.4 Poor output by the Drafter

In both the histogram-style experiments and the gridfont-level output, the Drafter made a number of extremely poor decisions. The following sections identify some of the reasons why those poor decisions were made.

Nearsightedness I: Global properties

An earlier epigraph noted that the environmentalist slogan, "Think globally. Act locally." describes the approach of the Drafter. The emphasis on local activity is particularly true in Coderack drafting, in which quanta are added to the gridletter one at a time, as though a pen were being moved around the grid, drawing as it goes. Each run of the Coderack makes a choice between which, if any, of the quanta adjacent to the current position of the imaginary pen should be added to the gridletter next. This decision is made without the expectation that the gridletter can be made satisfactorily complete with the addition of just one quantum (that expectation would be warranted, of course, if the gridletter were nearly complete, but will not usually be true when only a portion of the gridletter has been drafted). The decision is made, rather, with the goal of adding a quantum that brings the gridletter a step closer to completion. Because the decision is made when the gridletter is still incomplete, the Drafter is not able to take into consideration how the quantum (if any!) that is added will interact with subsequent drafting. Deciding which quantum to add next without worrying (yet) about how the finished gridletter might look gives the Drafter a kind of nearsightedness that causes it to make some poor single-quantum choices along the way — choices that, as later examples will make clear, look good to it in the short run, but that make it impossible for the Drafter to come up with a good finished gridletter.

Coderack drafting, however, can look at the big picture (that is, consider how future drafting, beyond the current step, might proceed) in one limited way. This comes about by means of the trait-progress sugar type. Each trait-progress codelet distributes sugar among the candidate quanta on the basis of a trait (such as height, weight, roof, floor, etc.). Each candidate quantum is given a fixed score of zero unless the role-filler, with the addition of that quantum, is obviously ruined with regard to the trait that it is considering. For example, with height, it gives a score of zero (neither reward nor penalty) to all candidates that do not make the height greater than the expected height, but a negative score (a penalty) to all those that do exceed the expected height. The reason why candidates that leave the role-filler one quantum shorter than the ideal height are scored better than those that leave the role-filler one quantum too tall is that a role-filler that is too short can be made taller later in the drafting process, whereas a role-filler that is too tall cannot be made shorter later (the Drafter drafts only by adding; never by removing). The trait-progress sugar type thus keeps Coderack drafting from making certain decisions that would lead to a flawed final gridletter.

However, the activity of trait-progress codelets is not enough to keep the near-sightedness of the Drafter's one-step-at-a-time approach from making some serious kinds of mistakes. This can be seen in an example illustrating how the roof trait (via trait-progress codelets) influences the drafting of an f-post. Figure 7.8 represents the situation facing the Drafter as it tries to decide which quantum it should draw as the very first step in the drafting of an f-post. The figure shows the starting point (marked "START"), the putative final point (marked "FINISH"), and the possible steps with which it can begin drafting, labeled by their quantum numbers. In this example, the continue-sugar score is the only one applicable, as no single quantum will rate highly as a one-quantum version of the entire role-filler. Suppose that the goal trait for roof is 'top' (the top of the grid), that the goal trait for height is 'tall' (baseline to top of the grid), and that curve-following sugar is about equal for all the choices but *19* about equally. In this case, the program has no basis for preferring *45* over *35* as the first step. Neither ruins the possibility of the role-filler eventually reaching a height of 'tall', nor does either ruin the possibility of the role-filler eventually having a roof at the 'top' level. The problem is that beginning with quantum *35* makes it extremely unlikely that the role-filler will ever in fact get back up to the top line before it goes down to end at the 'FINISH' point — unless it sacrifices the goal of looking like an f-post. Nothing in trait-progress calculations lets the Drafter see far enough ahead to detect the problem.

Two possible ways of modifying the program were tried, and one was implemented, but neither fully fixed all the problems. The first was to add trait-met sugar to the continue-sugar score; this awards positive scores to candidates that, when added to the role-filler, make the role-filler meet the desired trait *immediately* (not potentially, after the addition of other quanta, as with trait-progress). That change to the program would fix the *45–35* problem, by favoring quantum *45* over quantum *35*, because the former meets the roof trait of 'top' immediately. However, it breaks more situations than it fixes, including other f-posts. For example, with a width goal trait of half-wide, quantum *16* would never be chosen over quantum *35* as the first step, because the latter gives the role-filler the correct width immediately, and the former does not. Attempts to keep the beneficial aspects of this change but eliminate the harmful ones by weighting the different kinds of sugar type carefully seemed to indicate that no simple solution exists.

The second attempt at a solution was implemented, but only mitigates the problem, and does not solve it. This solution was based on the observation that the role-filler will probably eventually include the 'FINISH' point (or some point near it). As a result, it can be said, even before the first quantum is drafted, that the height will probably end up at *least* as great as the difference between 'START' and 'FINISH'. Based on this observation, the solution is to pretend that 'FINISH' is already in the role-filler, and to add trait-progress with the trait-met sugar type, as calculated for the role-filler plus 'FINISH', in the calculation of the continue-sugar score. With that solution, quantum *45* loses its categorical advantage over quantum *16* with respect to a half-wide width, because with the addition of 'FINISH', the role-filler is already half-wide with the addition of either of these candidates. In addition, the same strategy gives quantum *45* the advantage it should have over quantum *35* with respect to height, because it gives the role-filler its proper height value. The problem with the solution is that it only works when the first step plus (including) the 'FINISH' gives the role-filler precisely the desired value for the given trait. It is, in fact, destructive in other cases. Considering 'FINISH' already to be part of the role-filler removes the advantage that some candidates should have over others.

The solution finally employed was to make trait-progress the weighted sum of the first definition offered for trait-progress — a score that rates a candidate based purely on whether or not it ruins the role-filler with regard to the trait passed in as a parameter — and the calculation of trait-met for the role-filler plus the 'FINISH' point. The former is given a much greater weight than the latter; this approach seemed to reduce, but not eliminate, the nearsightedness that is inherent in the approach of drafting one quantum at a time.

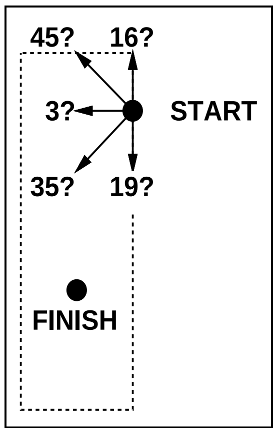

Figure 7.8: Looking ahead: The first step of an f-post.

Nearsightedness II: Burning Bridges

The inability of the Drafter to truly look ahead can cause problems in many ways. One is that an early decision, either of a quantum or of an entire role-filler, may limit the Drafter's options so that it cannot properly draft the rest of the role-filler, or other role-fillers for the Drafter, as the case may be, and end up with a good gridletter. Figure 7.9 shows two examples in which the Drafter has rendered one role-filler for a whole, and the role-filler has no intrinsic problems with it, except for the fact that it makes it impossible for the Drafter to finish the gridletter and do a good job of respecting both letter and spirit.

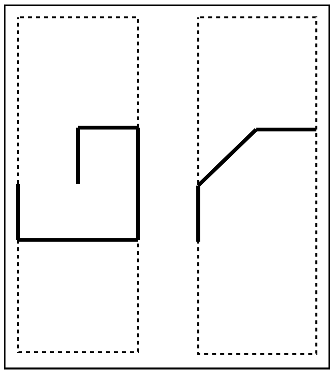

Figure 7.9: Best intentions: Two good parts that lead to bad wholes.

On the left is a right-bowl that the Drafter might render while attempting a 'p' in the style of Square Curl. This is squarish, incorporates the appropriate motif, and is a decent right-bowl, even if a little odd. The problem is that no left-tail can now be drawn to finish the 'p' and maintain the style. Beginning the left-tail from the lower left corner of the bowl and making it drop straight down from there would deviate from the Square Curl style. Beginning it from the upward-pointing tip on the left would force the left-tail to cross back through the bowl, somewhere, in order to get down to the descender zone. Beginning the left tail from the point in the center of the baseline might save the situation, but it is very unlikely that that point would be chosen as the starting point, because that reflects neither any norm for ‘p’ nor any norm violation common in Square Curl. The Drafter will thus almost always produce, at best, a marginally flawed version of what it was attempting, once it has begun with that bowl.

On the right is a forward-slash that might be drafted for an ‘x’ in the style of Benzene Right. There is nothing about this that makes it intrinsically a bad forward-slash, or intrinsically bad for Benzene Right. However, any possible rendering of the backslash that would complete the ‘x’ must either enter the ascender zone, which would seriously violate both letter and spirit — or feature a diagonal quantum in the backslash direction, which would violate the spirit of Benzene Right — or do something even more radical and contradictory to both the letter and the spirit. There is no reasonable test that could be performed on the forward-slash shown that would reveal its problematic nature without considering how the forward-slash might finish the gridletter. There are countless possible examples of this phenomenon, wherein a decision that looks reasonable early on leads to problems later.

From bad to worse

Among the most striking examples of bad Drafter output presented thus far are the third, fourth, and fifth ‘l’s in Figure 7.3. These are very similar to one another, and uniformly bad! Apart from the fact that the Drafter can produce such terrible output a fifth of the time, it is even stranger that it should be so bad in the same way, over and over. As it happens, in the case of these ‘l’s, the cause is the same every time.

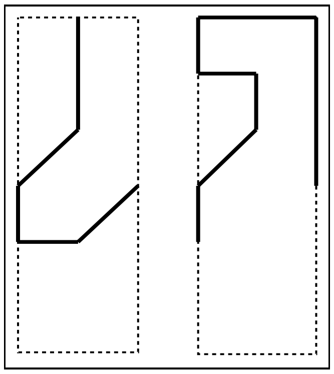

Figure 7.10: From bad to worse: Two ‘l’s, similar in their origins but very different in their execution.

Figure 7.10 shows a fine attempt at ‘l’ in Benzene Right on the left. On the right is one of the attempts gone horribly wrong that was presented earlier. The lead-up to the actual drafting of these gridletters was remarkably similar. They are attempts at the same role-set with the same goal style, down to the exact stylistic properties in the Thematic Focus. The difference began with the fact that the start point and the finish point were reversed. The Drafter decides at the beginning of each run at which end to start and at which end to finish, in order to give drafting more variety. The points that became the tips of the ‘l’ on the left were the starting and finish points chosen for both ‘l’s, though for the gridletter on the right, only one of them actually wound up being a tip.

The ‘l’ on the left was drawn from top to bottom, roughly counterclockwise. In general terms, the first steps added to the center-post role-filler’s height, and dropped from the starting point a bit nearer to the prospective finish point. When the current-point came to the top of the central zone, the strength of the Benzene Right motif began a course that started leftwards, then wound around to the right, covering more than half of a circuit of the central zone, and ended up at the preferred finish point.

Remarkably, reversing the start and finish, and beginning a new ‘l’ in this same style with the point where the ‘l’ that was just described ended leads, quite reliably, to something like the ‘l’ on the right. In this case, beginning in the middle of the right edge of the central zone, the first steps also prefer to close quickly the distance between the starting point and the preferred finish point (which is now above the starting point). The pathway leads up, and ruin is already assured. For reasons very like those in the example of the f-post mentioned earlier, the program is blind to the fact that heading up in the first steps leaves the desired floor trait (the basebar) unattainable without ruining the role-filler as a central-post. The Drafter heads up the right side of the grid, then turns and reaches the desired finish point. However, the role-filler has numerous problems: it is not wide enough, its left edge is too far right, and its floor is too high. Therefore, the step that led to the finish point has a terrible trait-met score, and is not accepted on the basis of its quit-sugar score. The drafting continues, and what would have been a poor ‘l’ for Benzene Right becomes an awful ‘l’ by any standards. One more step to the left fulfills the left-edge and width traits, but this step was not made on the basis of quit-sugar, since the upper left corner of the grid is not among the quit-points, and also since the floor trait remains unfulfilled. The Drafter continues “scribbling” through several more steps, on the way making some gestures towards the Benzene Right style, but without salvaging the ‘l’-ness.

The point of this example is to explain why some Drafter output is not merely bad but *awful*. In most cases, the cause is that a series of rather innocent decisions, some going a bit astray due to the nearsightedness that comes from Coderack drafting’s reliance upon local decision-making, leads to a point of no return, after which it is inevitable that the output will be seriously deficient.

Flukes and other luck

There are other sources of undesirable behavior in the Drafter. The nondeterministic nature of the Drafter means that *any* kind of fluke in drafting is possible, though most are unlikely. Many extraneous wiggles and glitches in the output stem from the fact that the Drafter was designed to produce variety in its output. Sometimes, needless to say, the results of this striving for variety are very poor.

The case of ‘o’ is a good one for demonstrating the tradeoff between variety and blundering; it is impossible to elect the first without getting the second. Figure 7.11 shows many (though by no means all) of the possible ‘o’s that can be drawn on the Letter Spirit grid. This is, in fact, a comprehensive list of the sixteen ‘o’s that fit in the central zone and that, in each of the four corners of the central zone, either “square off” the corner, by employing the two quanta that make a right angle in that corner, or “cut off” the corner, by taking one diagonal quantum to get past the corner. The sixteen ‘o’s in the figure show all the permutations that either square off or cut off each corner.

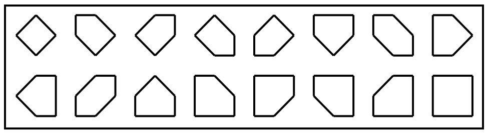

Figure 7.11: Sixteen ways of drawing an o-ring.

Given a style in the Thematic Focus, for each corner, there is a preference regarding whether the corner is cut off or squared off. In the case of Standard Square, the preference is that all of them be squared off. In the case of Hint Four, the preference is that all of them be cut off. The two Benzenes, Boat, and House have their own preferences, and so on.

The variety–blundering tradeoff forces the programmer to decide how much of the former is necessary and how much of the latter is tolerable. The programmer’s decision can be thought of in terms of a hypothetical parameter in the Drafter that decides, at each corner of an ‘o’, for instance, what the probability is that the style’s preferred behavior (cut off or square off) will in fact be taken at each corner. To have a maximum of variety with a minimum of blundering, the ideal would be to have a high probability of the entire ‘o’ matching the style’s preferences, but also that at any given decision point, there would be a modest probability that the Drafter would explore the less-favored alternative. In this way, a set of several attempts would produce variety, but also would produce the one “right” answer that agrees with the style’s preference, in case that truly happens to be right.

Unfortunately, if the means of controlling the amount of variety to be utilized in decision-making is to be reduced to a single parameter, then the two goals are incompatible. If the probability of the Drafter investigating an intriguing possibility at one location is p, then the probability of the entire ‘o’ following the style’s preference is $(1 - p)^4$. If p is 50%, then there is a reasonable chance of exploring variety, but the probability of the ‘o’ matching the style’s preference is only 6.25%. To make the probability of the ‘o’ matching the style’s preference 50%, then p must be a mere 16%. The actual program does not have one parameter quite like the hypothetical p described above, but something very much like it emerges from the complexity of the program; the solution that was chosen was to allow a fair amount of both variety *and* of blundering. Fortunately, the Drafter’s blundering, as well as many of its other shortcomings, can be redressed by the systematic revision afforded by the combined power of the Examiner and Adjudicator, to be discussed in the following chapter.

7.4 Conclusion

The Drafter models certain elements of human gridletter rendering. Borrowing shapes from other gridletters is a strategy that people clearly employ from time to time, and the Drafter models it successfully. Coderack drafting is meant to capture the way that multiple pressures on a subcognitive level can collectively lead to a decision on the cognitive level. The kinds of pressures it includes in its calculations are also evidently ones that influence human gridletter rendering. However, it is not possible to show that the match between the program and people is very tight, for reasons that are revisited below.

The Drafter was designed to be one module in a larger architecture and therefore its output must be judged in that light. Many of its attempts result in output of poor quality, but this is a fairly direct consequence of the fact that it was designed to produce variety in its output. Decreasing the variety (by changing several internal parameters appropriately) could raise the quality of its output, and make this more like a typical AI model, along the lines of those surveyed in Chapter 4. An essential goal of the Letter Spirit project, however, is to test an architecture that has variety in its tentative output, so that the program itself can review its own work.

Not all of the Drafter's shortcomings are explained by the mechanisms that introduce beneficial variety into its output. It is also nearsighted in a number of ways. When a run goes awry, the Drafter often carries on needlessly, creating a shamelessly messy scribble as its attempt at a gridletter. Ways of having the Drafter itself detect this kind of situation and quit immediately rather than waste extra running time (and "embarrassing" itself further) were investigated, but most options seemed to require that the Drafter have an ability to see the bigger picture, beyond the local level. It is, of course, the absence of any such capacity that leads to many of its errors in the first place. Although it would be a major undertaking to give the Drafter a view of the big picture while it makes the individual quantum-sized steps, that is certainly a worthy goal for future work.

The Drafter is, for reasons that have already been noted, a creator that creates while largely blind to the quality of its output. Letter Spirit was implemented under the assumption that the Drafter's output would be accepted only after it had been reviewed by the other modules, and for that reason, Drafter output that has not been subjected to that process of review is often deeply flawed. Unlike the Drafter, a person can review tentative output — on some level — while it is still only a mental representation, and can then approve of, revise, or completely censor the output based upon the review. The subjective experience argues forcefully for this possibility, and there is also neuropsychological support for it. A strong case has been made for the existence of Kalman filters, mechanisms that review the mental emulation of the results of contemplated behavior before it actually takes place [Grush 1995]. Because of the seemingly ubiquitous role of revision and review in human creativity, the Drafter cannot be held up as a model of any externally-observable human behavior. Therefore, this chapter does not attempt to verify the correctness of the approach taken by the Drafter by comparing its output to that of people in some experimental condition; it seems unlikely that people ever produce work without reviewing it — at least while it is in some earlier internal stage of formulation. Thus, while it has been enlightening to discuss the isolated Drafter's output, the ultimate test of the Drafter is the quality of the output of the Letter Spirit program as a whole, which will be presented in the next chapter.

The three modules reported in these last three chapters all manage some impressive behavior, though each has clear shortcomings that should be addressed in future work. The most important goal of the work reported in this thesis is the investigation of how those three modules, imperfect though they may be, can be coordinated into an integrated model of human creative behavior. With the characterization of the three modules, in their current implementations, now complete, the discussion can at last turn to the implementation and performance of the top-level Letter Spirit program.

CHAPTER EIGHT

Letter Spirit

I'm not a very good writer, but I'm an excellent rewriter.

— James Michener

8.1 Introduction

The Examiner, Adjudicator, and Drafter are all programs of considerable sophistication. The top-level control of Letter Spirit ties them together into a relatively detailed model of human creativity in the domain of gridfont design. Its task is to carry out "the central feedback loop of creativity," in which everything the program creates is inspected for quality before it is deemed worthy of inclusion in the final output. In some versions of the program, it can also carry out a related task of progressively evolving its own sense of a goal style — the goal style that is guiding gridletter creation in the current run.

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

8.2 Implementation

8.2.1 An overview of processing in Letter Spirit

The bulk of Letter Spirit's code and complexity lies in the three modules described in the previous three chapters. The simpler top-level program that coordinates the modules' activity consists of two phases.

The first phase categorizes and analyzes the seed letters given to the program to build up a representation of the style that they have in common.

The second phase of the program is a loop, in which a letter category is selected (nondeterministically, but favoring those categories that do not yet have a good instantiation in the developing gridfont), and then the Drafter renders a gridletter that, ideally, incorporates the goal style and instantiates the letter category. The Drafter's attempt is shown to the Examiner and the Adjudicator, and if the attempt is deemed to be the best version thus far for that category, as determined by the scores that the Examiner and the Adjudicator generate, then it is kept as the current version of that category in the gridfont. This loop runs many times, and as it does so, the quality of the gridfont should incrementally increase.

8.2.2 The Scratchpad: Memory for the top-level program

Letter Spirit's top-level control uses just one major memory structure that is not used by any of the modules — namely, the Scratchpad, which is the repository of the gridfont in progress. For each of the 26 letter categories, the Scratchpad stores a list of entries, one for each version of the gridletter that has been rendered as an attempt for it during the current run. Each entry consists of the quanta that made up the attempt and a list of scores rating the quality of the attempt. The scores range from 0 to 300, with low scores being the best (because they are based on temperature readings from the Examiner and the Adjudicator).

Initially, each letter category has stored for it only one attempt — the default of an "empty gridletter" containing no quanta. The default score for that null entry is a fraction of a point under 300. A score of 300, the worst possible, is reserved for attempts that are not recognized by the Examiner as members of the intended category. In all other cases, scores tend to be between 0 and 100. A gridletter presented to the program as a seed for a gridfont is given a perfect score of 0, so that it can never be replaced. At all times, the best entry thus far is given a distinguished position, and this attempt is placed in the Scratchpad display. This enforces a succession, throughout a run, in which a non-seed category remains blank until the first attempt for that category is recognized as a member of the category. At that time, the successful attempt becomes the new "champion" and is placed in the display. Thus begins a succession of champions, one for each category; this is the essence of each Letter Spirit run.

8.2.3 Other variables and parameters

A few variables and parameter settings control some important aspects of Letter Spirit.

Mentioned in the Drafter chapter, a (virtual) randomness knob is slowly turned upwards throughout a run. This knob affects how randomly the Drafter makes choices throughout a run. As a result, variety increases at the cost of additional and more serious blunders. Late in a run, it is likely that a decent version of each category has already been created — one that is the best of several attempts. Therefore, the cost of a blunder is small (it will not dislodge the best version already created), while the potential rewards of variety (that a rare but high-quality version of a letter category is produced) are high.

Another knob that can be adjusted is the degree to which style is allowed to influence drafting. This knob is turned down when the Drafter attempts to render a gridletter for which all previous attempts during the run have been rejected by the Examiner; the more attempts that have been made and rejected, the lower the knob is set. The motivation is to decrease the importance of style if it is a hindrance to the production of *any* acceptable rendition of the category. If this knob did not exist, then if Letter Spirit were to fail to find an acceptable version of one letter category, repeated attempts (and failures) directed at that single letter category would tend to monopolize the program's efforts throughout a run.

These two knobs set the degree to which the program focuses its behavior, or "de-focuses" it (promoting greater variety), and together they play a role quite like the role that temperature plays in many FARG models.

Other parameters can be changed by the user from run to run, to test different variations on the basic strategy. Which promotion strategy (explained below) to use is one such parameter. How long to let the program run (in terms of total number of Drafter attempts) is another — this is set at 300 for all runs in this chapter, unless otherwise noted. Finally, which gridletters, if any, are given to Letter Spirit as seeds varies from run to run and, obviously, this has enormous influence over Letter Spirit's behavior.

8.2.4 Processing in Letter Spirit

The previous chapter vividly demonstrated that the Drafter does not have a high rate of success in creating excellent versions of letters on the first try — versions that capture both letter and spirit very well. The success (such as it is) of the Letter Spirit program comes, rather, from the fact that many versions are drawn for each letter category, with only the one that is rated best by the program making it into the final gridfont.

The first, seed-handling phase of Letter Spirit is quite simple. Each seed is run past the Examiner, which determines its letter category. The Adjudicator then runs on the seed, using the Examiner's parsing of it. As the Adjudicator runs over the entire set of seeds, it builds, in the Thematic Focus, a representation of the style to be used in designing the remaining letters. In addition, the seeds are included as members of the gridfont, each in its appropriate letter category.

The second phase, in which the program searches for ever-better versions of each non-seed letter category, is the real workhorse of a Letter Spirit run. At the beginning of the second phase, all non-seed letter categories are the subject of a series of attempts to improve the gridfont, one attempt per pass through the loop. First, a letter category is picked as the one to benefit from the next attempt. This is done nondeterministically, but is weighted by the score for each category's current champion. A category with a champion rated at 80 has twice the chance of being picked as a category with a champion rated as a 40. Seeds have no chance of being selected, being automatically assigned a rating of 0. Letter categories that have not yet been rendered successfully even once have a score of slightly under 300, which means that they will receive the most attention, probabilistically speaking, until every category has at least one acceptable rendition in the display.

When a category has been selected, the Drafter renders a gridletter for that category, drawing on the style in the Thematic Focus, as was described in the previous chapter. This attempt is run past the Examiner and the Adjudicator, and is assigned a score that sums the way each of those modules rates the gridletter. If the attempt is the best version thus far for that category, as determined by its score, then the attempt is kept in the gridfont as the current champion for that category. The program does not, however, discard "defeated champions"; rather, it stores all attempts made for each letter category during a run in the Scratchpad, along with all the Examiner and Adjudicator ratings for each attempt.

If the new gridletter's total score is close enough to the score of the previous-best gridletter for the given category, then both the new gridletter and the current champion for that category are evaluated one more time, by both the Examiner and the Adjudicator, and all subsequent comparison is based upon the mean of each gridletter's two (or more) scores. This may cause the champion to receive a worse score, particularly if it is a gridletter that can be seen as a member of more than one different category by the Examiner. A gridletter such as this can receive an excellent rating on one pass but have its overall score ruined when a subsequent pass through the Examiner identifies the gridletter as a strong member of another category as well. In any event, after the new attempt and the current champion have both undergone this independent review and received an additional score, the new champion is determined by comparing the scores of *all* the attempts that have been made for that letter category in the current run. This means that a previously-rejected attempt that was stored away can wind up being promoted as the new champion, though this happens rather infrequently.

The loop runs many more times than there are categories, so usually multiple attempts are made for each category. When the program quits (after it reaches the predetermined number of 300 total drafting attempts, or — in most situations unlikely — when every category has a version in memory that has been promoted to seed status), the final version of the gridfont consists of the set of the best attempts it has made for each category, plus the original seeds.

8.2.5 Promotion

And have ye not read this scripture:
The stone which the builders rejected is become the head of the corner.
— Mark 12:10

One of the most important goals of the Letter Spirit project is to capture the way that a style can evolve during the creative process. Journeys do not always end at their intended destination, and the "goal" style for the last letter designed in a typeface may be quite different from that of the first letter. In Letter Spirit, the process of evolving a style is modeled by the *promotion* of gridletters designed by the program to the same status as that of the seeds that are used at the beginning of a run to determine the initial style.

Promotion introduces a host of issues. As a test of the principles behind review and revision, the purest test is not to allow evolution of style, and to keep the goal style fixed throughout a run. However, in order to explore as broad a range as possible of the issues at hand, three variations on Letter Spirit — one banning promotions and two allowing it — have been tested.

The most conservative version of Letter Spirit does not allow style to change at all. In such no-promotion runs, seeds are presented and incorporated, one at a time, into the representation of the goal style for the current run. They are also added to the Scratchpad. Once all of the seeds have been handled, the goal style is effectively immutable, and all further work is aimed at finding better versions of the non-seed letter categories, aiming always at imbuing new letters with the initial goal style.

In the two versions of Letter Spirit employing promotion, newly-minted gridletters can alter the Thematic Focus and the Library. Promotion takes place only with gridletters that have already been evaluated by the Adjudicator (as well as by the Examiner); at promotion, the gridletter and its parsing by the Examiner are added to the Library, and the gridletter is used to update the Thematic Focus. Two different promotion strategies were tested. "Easy promotion" features a low threshold for determining when promotion may take place, while "tough promotion" allows it more rarely.

In all three versions of Letter Spirit, once a gridletter has been promoted to seedhood (whether as an original seed or as a later promotion), it is added permanently to the Scratchpad and is not subject to further revision. This does not allow one strategy that a human designer might use — namely, finding a new goal style, based on some mid-gridfont creation, and throwing out some or all of the gridfont created up to that point. That kind of radical style evolution (literally "radical", if one considers the gridletter or gridletters that triggered the change to be a root from which a new style can grow) may be explored in future work on Letter Spirit.

8.3 Performance

Messrs. K and H assure the public their production will be second to none.

— John Lennon and Paul McCartney, Being for the Benefit of Mr. Kite

The goal of this project is to shed light on key mechanisms of human creativity through a computer model. The performance of Letter Spirit illustrates how close the current implementation is to the reality of human creativity, and how far from it. The model is based on certain assumptions and simplifications, and so the performance of the program provides important insights as to which assumptions are correct, which simplifications are unwarranted, and what directions future research should take.

The gridfonts in this section come from a few variants both on the Letter Spirit program and on the task assigned to them. Two test sets of human-designed gridfonts are used to seed Letter Spirit runs. From each gridfont, a subset of 5 gridletters is used as the seeds. In most cases, the five seeds are the BCEFG gridletters from that gridfont, although one of the variations uses a different set of letter categories while keeping the choice of gridfonts constant.

The SMALL set of gridfonts, introduced in Chapter 6, consists of Benzene Right, House, Shorts, Snout, and Standard Square. The LARGE set of gridfonts consists of all 15 of those 23 gridfonts introduced in Chapter 1 that can be recognized reasonably well by the Examiner; the LARGE set is SMALL plus Benzene Left, Boat, Close, Double Backlash, Flournoy Ranch, Hint Four, Hunt Four, Sabretooth, Slash, and Weird Arrow. The remaining gridfonts from the set of 23 cannot be recognized reliably enough by the Examiner for their BCEFG gridletters to be used as seeds for Letter Spirit runs.

8.3.1 No promotion

Letter Spirit runs without promotion demonstrate the review-and-revision strategy in its purest form. Without promotion, the goal style is fixed throughout a run, and the output shows how (and how well) the program proceeds to create a gridfont in that fixed style. Figure 8.1 shows the output from one Letter Spirit run each on the SMALL set of gridfonts.

At the risk of "letting the inmates run the asylum", the Examiner and Adjudicator's ratings of the non-seed gridletters of these gridfonts are included in Table 8.1's assessment of these gridfonts. These five numerical ratings are given for every Letter Spirit gridfont reported in this chapter.

Exam is the mean Examiner score from three runs on each of the 21 non-seed gridletters. *Norec* is the number of times, in three tests per gridletter, that the Examiner failed to recognize a gridletter as a member of its intended letter category.

Adj is the mean Adjudicator score from three runs on each of the 21 non-seed gridletters. In these tests, the Thematic Focus was first cleared from the Letter

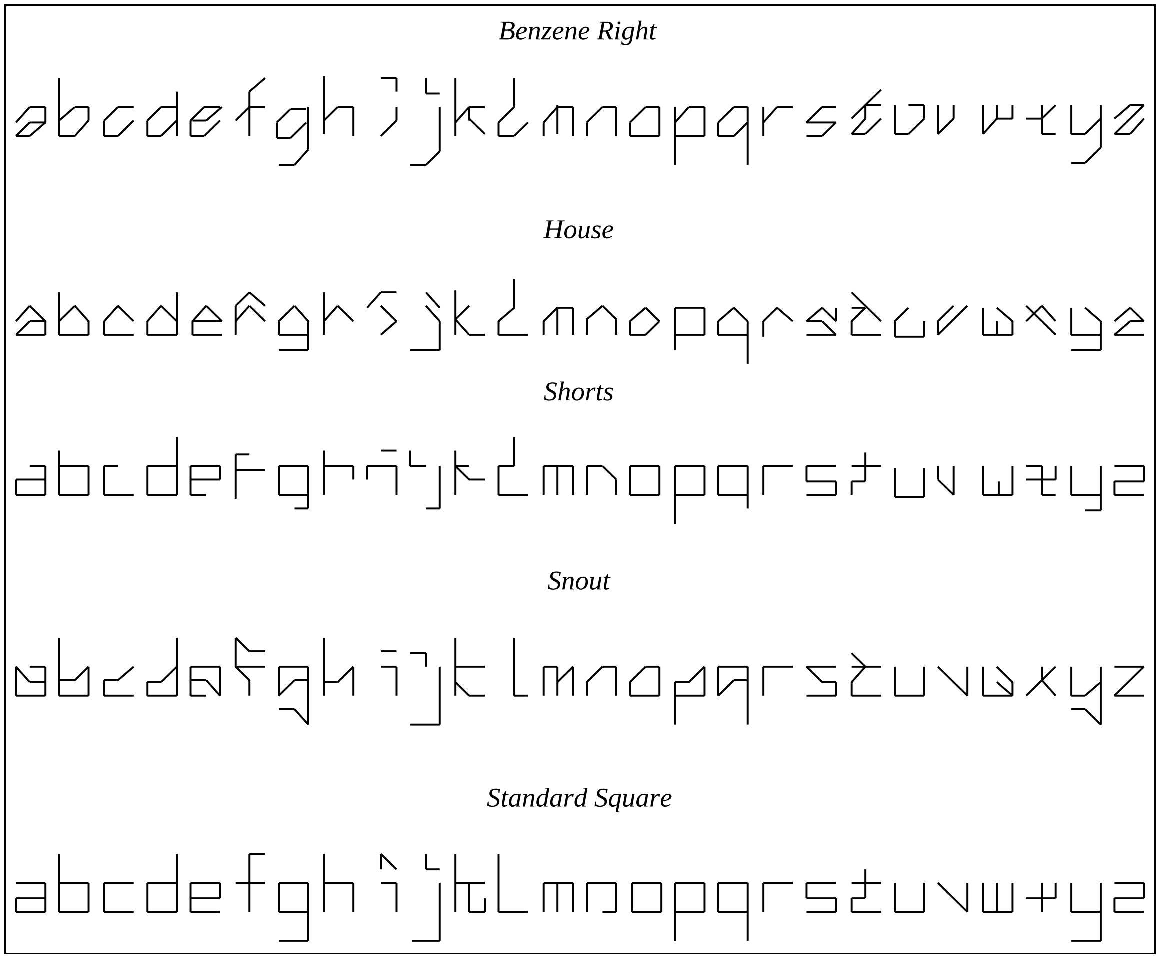

Figure 8.1: Letter Spirit output: No promotion.

Spirit run that produced the gridfont, and then was re-trained on BCEFG. (The re-training was intended to reduce the extent to which the particulars of one run, with regard to the way that the style of the seeds was represented in the Thematic Focus, might lead to artificially high agreement between the gridfont and the post-production assessment thereof.) Each of the Adjudicator tests followed one of the three Examiner tests mentioned above. If an Adjudicator test's corresponding Examiner test did not lead to successful recognition of the gridletter, then the score resulting from that Adjudicator test was not included in the mean.

Same is the number of non-seed gridletters that Letter Spirit designed and that were exactly the same as the version of the same letter category in the human-designed gridfont from which the seeds were taken. One can say that a high number of gridletters in common with the original font indicates that the program did a good job of latching onto the human designer's intent, and mimicking that in the non-seed letter categories. This is one kind of success. On the other hand, finding exactly the same versions of those letters is not necessary for performance to be good. It is possible that the program will create some gridletters that are different from those of the human designer and that are still good, perhaps sometimes better than those that the human designed.

Finally, *Exam+Adj* is simply the sum of the Examiner and Adjudicator scores. This could be taken as the overall rating of the gridfont, although, since it is a rating of Letter Spirit's work *by* Letter Spirit, the objectivity of these scores should not go unquestioned.

	Exam	*Norec*	*Adj*	*Same*	*Exam+Adj*
Benzene Right	14.9	0	29.5	3	44.4
House	14.7	0	32.5	3	47.2
Shorts	24.7	3	11.3	2	36.0
Snout	12.7	1	47.6	3	60.3
Standard Square	8.0	0	21.3	12	29.3

Table 8.1: BCEFG seeds, no promotion.

Letter Spirit's output in these five tests is dramatically superior to the raw, unrevised Drafter output seen in Figure 7.7. Almost all gridletters are recognizable as members of their intended letter category. Most of them somehow represent their intended style, although not equally well for all individual letters or styles. Casually scanning each gridfont from letters ‘h’ to ‘z’ (past the seeds, which are human-designed), one sees a general similarity that shows up in most, but not all, members of these gridfonts.

Shorts and Snout are not handled particularly well by the Adjudicator (see Chapter 6), and so it is probably not surprising that they are not handled particularly well by Letter Spirit. Most of the gridletters in Snout that appear successful came from whole-gridletter borrowing, and thus were created without the involvement of the Thematic Focus. The key observation is that Letter Spirit *kept* the borrowed successes, because the Adjudicator gave them good ratings. The Thematic Focus's representation of style was inadequate for *drafting* in the Snout style, but not for *recognizing* the Snout style. This is a good example of how the loop architecture can make up for shortcomings in one area with strengths in other areas.

A few specific examples in this output illustrate key properties of Letter Spirit behavior. For several ‘l’s (note especially the Benzene Right ‘l’), Letter Spirit produced a gridletter that undeniably suited letter and spirit, in a way that integrated each gridfont's definitive motif to a far greater extent than in the original human-designed versions of these gridfonts. This is evidence of both acumen and also literal-mindedness on the program's part. A human designer is freer to take an exceptional approach to a letter category that is intrinsically atypical, as ‘l’ is.

The ‘t’s in both Benzene Right and House show another way that Letter Spirit differs from people. These gridletters both have the same quirk: at the junction where the two role-fillers cross, collinearity makes it hard for a person to see that the program created the gridletter in such a way that the role-fillers really did cross. Collinearity of contiguous quanta is a strong force in determining how people parse a letter; Letter Spirit's strict decompositional approach allows it to ignore this, which leads it to accept output that is weird to the human eye, and not in a way that benefits most conventional styles.

One characteristic of runs without promotion is that overall coherence can be compromised. The dots over ‘i’ and ‘j’ provide a clear example of this. For each of the five gridfonts here, the two dots are different. Without the prospect of either the ‘i’ or ‘j’ adding its dot to the Library so that the other might borrow it, the two dots can end up identical only by means of a bit of luck. This is one of many ways that an opportunity for coherence was missed because none of the letters created was made part of the definition of style.

Finally, the advantage that Letter Spirit has in handling squareness manifests itself again. Most (12 of 21) of the gridletters that Letter Spirit designed for Standard Square were exactly those that are found in the original human-designed gridfont, and some of its other work (note the ‘z’) is an impressive implementation of squareness; given the BCEFG seeds, there is no reason to suspect that standardness should take precedence over squareness, so pure squarishness is impressive. On ‘v’, the program relented and allowed diagonality, because it is essentially impossible to have a ‘v’ of which the Examiner approves without diagonal quanta in it. This required twenty attempts at ‘v’, with Letter Spirit all the while lowering the setting of the knob that determines the importance of style, until squareness was almost turned off and a relatively generic ‘v’ was accepted.

This discussion has not exhausted the issues raised by this output, but examples of other variations on Letter Spirit offer more examples of the same issues, as well as of numerous other issues.

8.3.2 Widespread promotion

The way that enabling promotion changes the character of Letter Spirit is a matter of great interest. Figure 8.2 shows the work of Letter Spirit on SMALL’s BCEFG seeds, using easy promotion. Table 8.2 shows the program’s ratings of its own work.

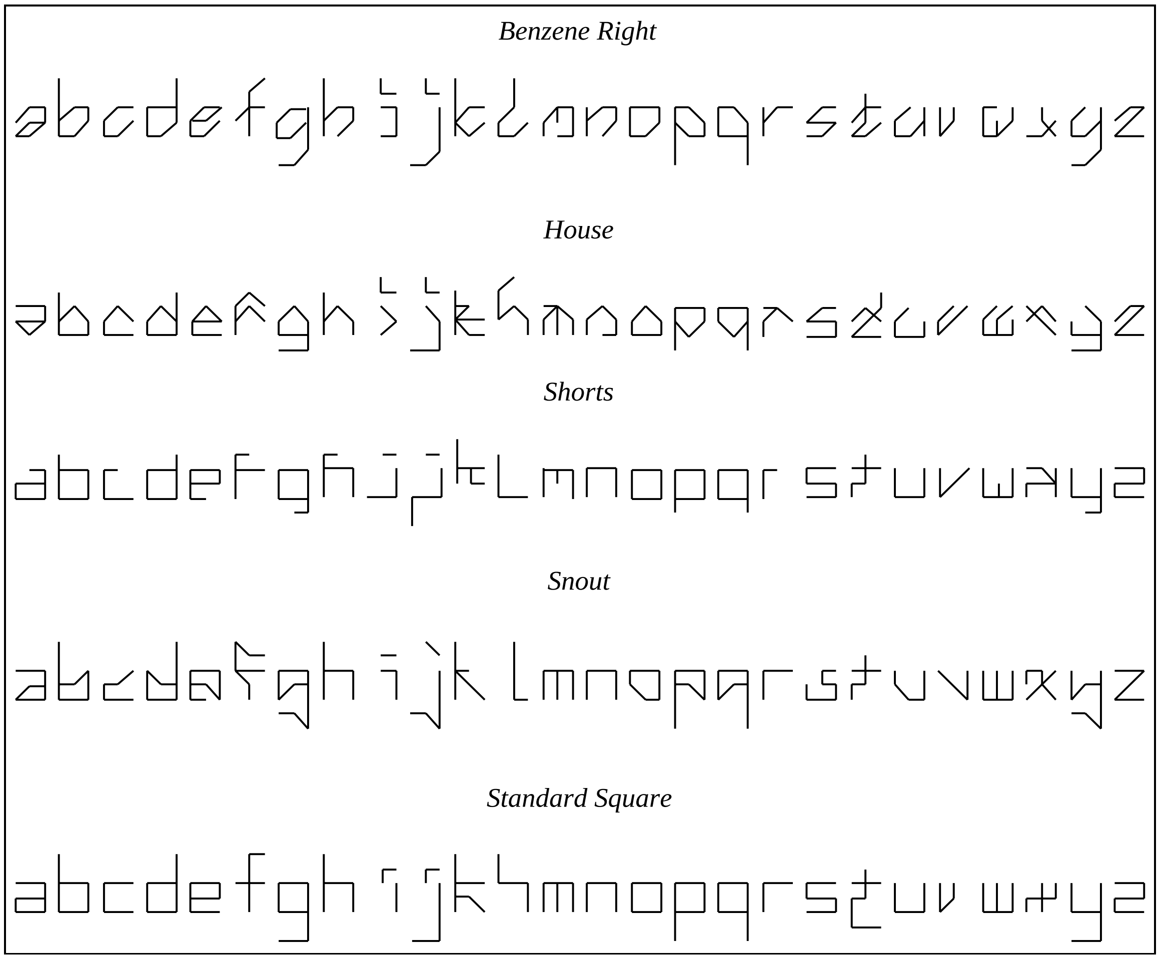

Figure 8.2: Letter Spirit output: Easy promotion.

A broad comparison between no-promotion runs and easy-promotion runs, based on the data in their respective self-rating tables, suggests that they are not very different. A specific way in which they do differ is in the dots over 'i' and 'j', where easy promotion led to the two dots' being identical in four of the five gridfonts. Coherence can be quirky; the 'm' and 'w' in Benzene Right both deviate from the main Benzene Right motif, but in a similar way. This is because one was drafted first, then promoted despite the discrepancy from the ideal, and then the other was borrowed from the first by whole-gridletter borrowing. The same phenomenon arises with the Benzene Right 'o' and 'd', and elsewhere.

The copying of quirks is exacerbated by the creation of them. Several gridletters in House seem to belong in Boat — that is, they point down rather than up. All three examples of this ('a', 'p' and 'q') were the result of whole-gridletter borrowing. The problem is that easy promotion is lenient in accepting such creations, not only as the best version thus far of their letter category, but as worthy of seed status. This makes their acceptance into the gridfont irrevocable, whereas in no-promotion runs, the creations like these might be accepted tentatively, but would be replaced if the program ever got around to creating versions that pointed the "right" way. This is clearly a problem with easy promotion; because whole-gridletter borrowing tends to involve post-and-bowl letters most, it can be said that Letter Spirit needs to watch its 'p's and 'q's. It could probably best be solved by a more complex regime of acceptance and promotion that allowed promoted gridletters to be demoted (and with them, letters that had been inspired by them). Nevertheless, tough promotion alleviates the problem to a degree.

	Exam	*Norec*	*Adj*	*Same*	*Exam+Adj*
Benzene Right	14.0	0	34.5	4	48.5
House	24.9	2	32.9	4	57.8
Shorts	18.4	1	15.4	6	33.8
Snout	8.4	0	48.5	3	56.9
Standard Square	15.7	1	17.4	13	33.1

Table 8.2: BCEFG seeds, easy promotion.

8.3.3 Limited promotion

Tough promotion balances the extremes between no promotion and easy promotion (though, as the previous section demonstrated, the two extremes did not lead to terribly drastic differences in the kind of output). This setting is used as the showcase mode of Letter Spirit — the *real* Letter Spirit, if only one promotion scheme is allowed to be considered "the" Letter Spirit. Consequently, this version of the program was run on the LARGE set of gridfonts, which allows for a comprehensive survey of how the program handles different kinds of styles. The next three figures show the output of these 15 runs.

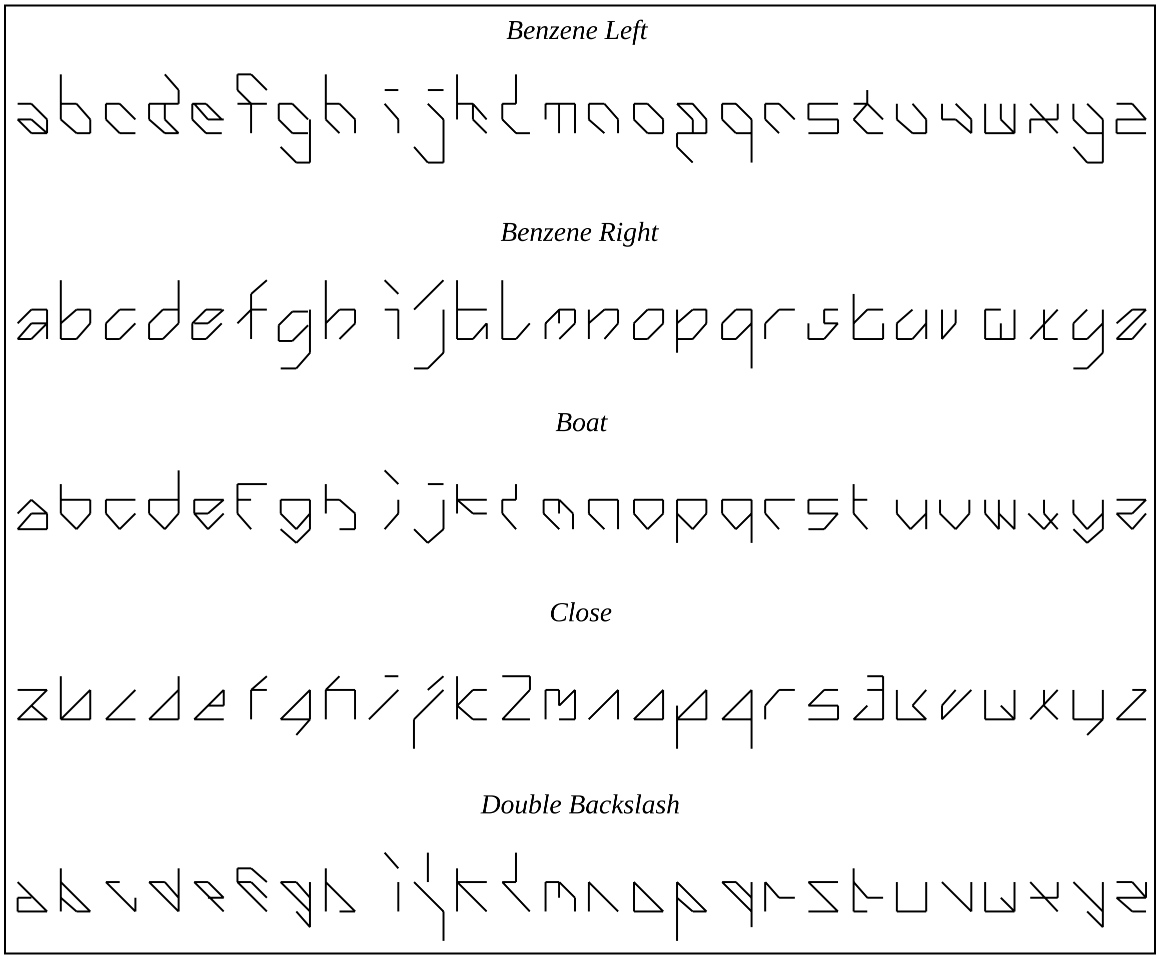

Figure 8.3: Letter Spirit output: BCEFG seeds, tough promotion.

One might expect performance with tough promotion to be somehow intermediate in nature between the no-promotion and easy-promotion conditions. This is generally true and most gridfonts are captured fairly well. Table 8.3 gives the self-ratings.

A comparison among the three promotion strategies' performance on SMALL

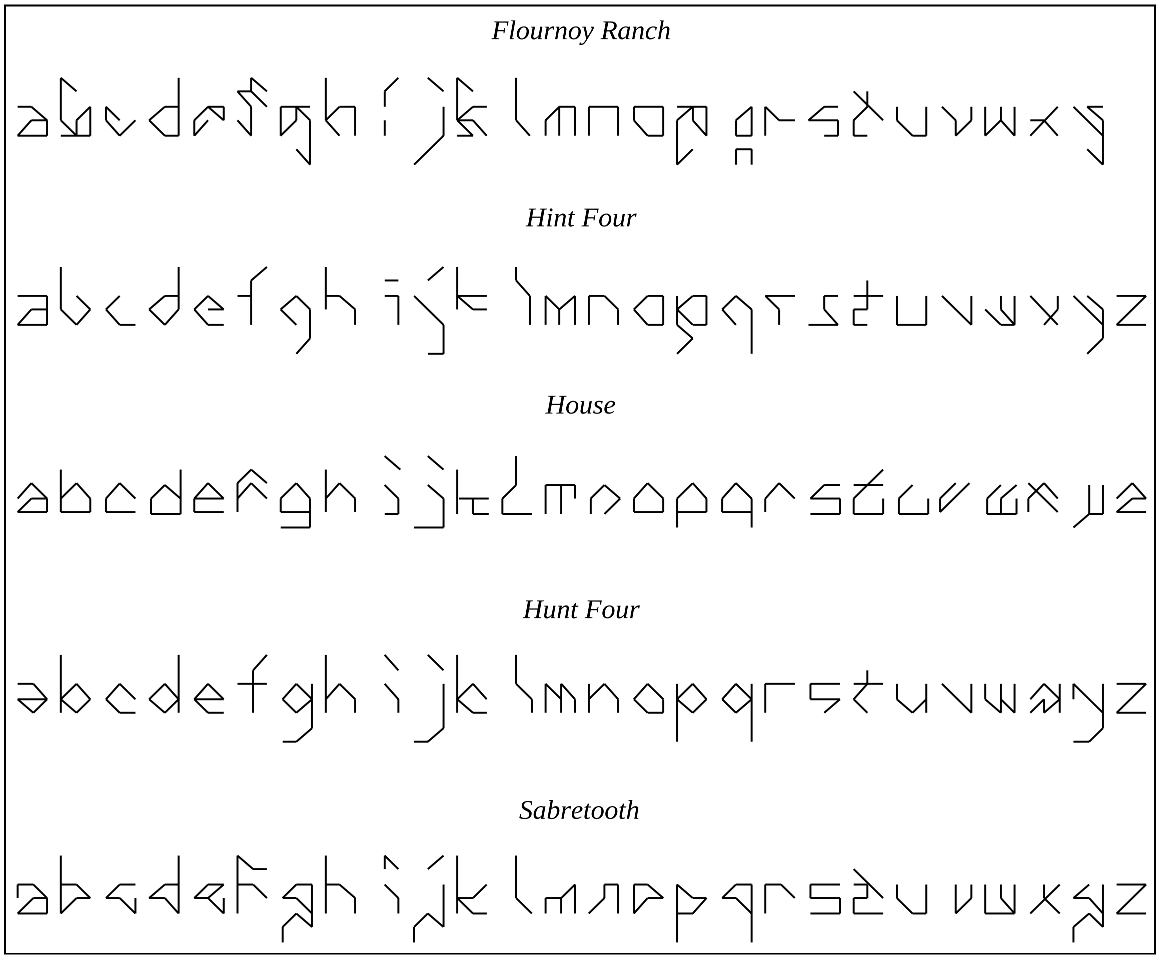

Figure 8.4: Letter Spirit output: BCEFG seeds, tough promotion.

(Table 8.4) shows that tough promotion led to the best overall self-ratings (which is why it was chosen as "the" Letter Spirit). It had the best rating in all of the categories except the Examiner's rating of letter category. However, the differences between the output produced by the three strategies is fairly subtle, and should not be overemphasized.

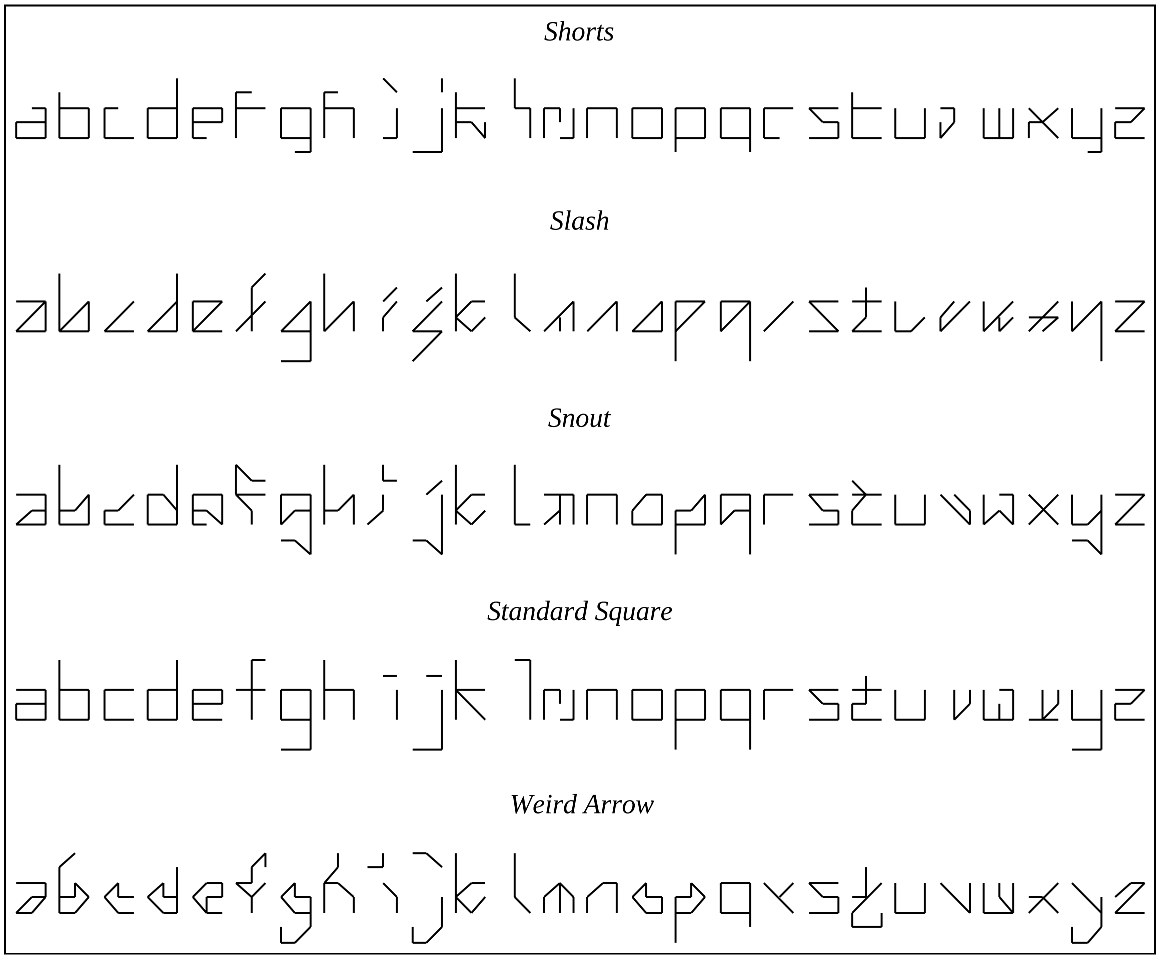

Figure 8.5: Letter Spirit output: BCEFG seeds, tough promotion.

8.3.4 Performance — by gridfont and by module

One of the most interesting aspects to study in all the Letter Spirit work to date is the extent to which different styles can be handled by each module; this is reported, as near as is possible, in Table 8.5. This table reproduces the Examiner's percentage of correct answers from the large test discussed in Chapter 5 and the Adjudicator's percentage of correct answers from the multiple choice test reported in Chapter 6. There is no column for the Drafter. The various parts of Letter Spirit are tied together in ways that make them hard to test in isolation, and, as in Chapter 7,

	Exam	*Norec*	*Adj*	*Same*	*Exam+Adj*
Benzene Left	15.1	0	26.1	7	41.2
Benzene Right	18.6	1	26.1	9	44.7
Boat	16.0	0	31.9	8	47.9
Close	21.1	1	37.4	4	58.5
Double Backslash	28.1	3	25.3	3	53.4
Flournoy Ranch	17.2	0	49.1	1	66.3
Hint Four	13.4	0	55.0	1	68.4
House	14.7	0	33.4	6	48.1
Hunt Four	10.5	0	46.8	7	57.3
Sabretooth	10.0	0	49.3	2	59.3
Shorts	16.6	0	11.7	3	28.3
Slash	16.9	0	31.1	5	48.0
Snout	14.7	1	49.2	2	63.9
Standard Square	15.0	0	10.0	10	25.0
Weird Arrow	14.1	0	48.7	1	62.8

Table 8.3: BCEFG seeds, tough promotion.

	Exam	*Norec*	*Adj*	*Same*	*Exam+Adj*
None	15.0	0.8	28.4	4.6	43.4
Tough	16.5	0.4	25.4	6.4	41.9
Easy	16.3	0.8	29.7	6.0	46.0

Table 8.4: Evaluation of promotion strategies.

the Drafter's performance is not rated quantitatively. However, a rating of how well the full Letter Spirit program handled each gridfont in the aforementioned tough-promotion tests manages to get at the Drafter performance, indirectly. The *LS* column shows the combined Examiner/Adjudicator score from those tests; here, however, it is subtracted from 100 so that, like the other two columns in this table, high scores correspond to good performance.

The *LS* rating is used as an approximation of a test of Drafter performance. In sharp contrast to the *Exam+Adj* entries in Tables 8.1 through 8.3, the Examiner and Adjudicator scores that are being summed to create the *LS* entries in Table 8.5 are not the same as those appearing in the separate columns to their left. In Table 8.5, the Examiner and Adjudicator scores are based upon the entire human-designed gridfont.

	Examiner	*Adjudicator*	*LS*
Benzene Left	95.4	90.0	58.8
Benzene Right	99.6	80.0	55.3
Boat	97.3	83.3	52.1
Close	94.6	77.8	41.5
Double Backslash	83.8	94.4	46.6
Flournoy Ranch	76.5	66.7	33.7
Hint Four	88.8	29.4	31.6
House	94.2	75.0	51.9
Hunt Four	95.8	27.8	42.7
Sabretooth	81.9	33.3	40.7
Shorts	97.7	45.0	71.7
Slash	85.8	86.1	52.0
Snout	96.2	50.0	36.1
Standard Square	100	72.2	75.0
Weird Arrow	87.7	41.1	37.2

Table 8.5: Evaluation of gridfonts: Performance by module.

The *LS* score, however, is based upon the gridfont that Letter Spirit designed based upon seeds from the human-designed gridfont (all but five of which were created by the Drafter). Therefore, while the Examiner and Adjudicator scores in Table 8.5 measure the ability of those modules to perceive each human-designed gridfont accurately, the *LS* score reflects the quality of the Drafter's work. This measure is relatively indirect, but nonetheless manages to distinguish between gridfonts that the Drafter handles quite well from those with which it has particular problems.

A Venn diagram of relative success by module and by gridfont appears in Figure 8.6, showing which modules succeed with which of the 23 example gridfonts. Arbitrary thresholds were set for delineating what is considered "success": for the Examiner, 85%; Adjudicator, 65%; Drafter, 50 points in the *LS* column. It can be seen that a few gridfonts are handled relatively well (how well will be discussed shortly) by all three modules, and some are not handled well by any of the modules. There is also a pecking order among the modules, with the number of gridfonts handled by each of them cascading from one module to the next; the Examiner handles 13 gridfonts well, the Adjudicator 10, and the Drafter just 7. This descent is more a matter of logic than of the intrinsic behavior of the modules. It is explicit in the program, as well as being borne out by the previous data, that the other two modules depend upon the Examiner, in large part, and the Drafter, in turn, depends upon the Adjudicator.

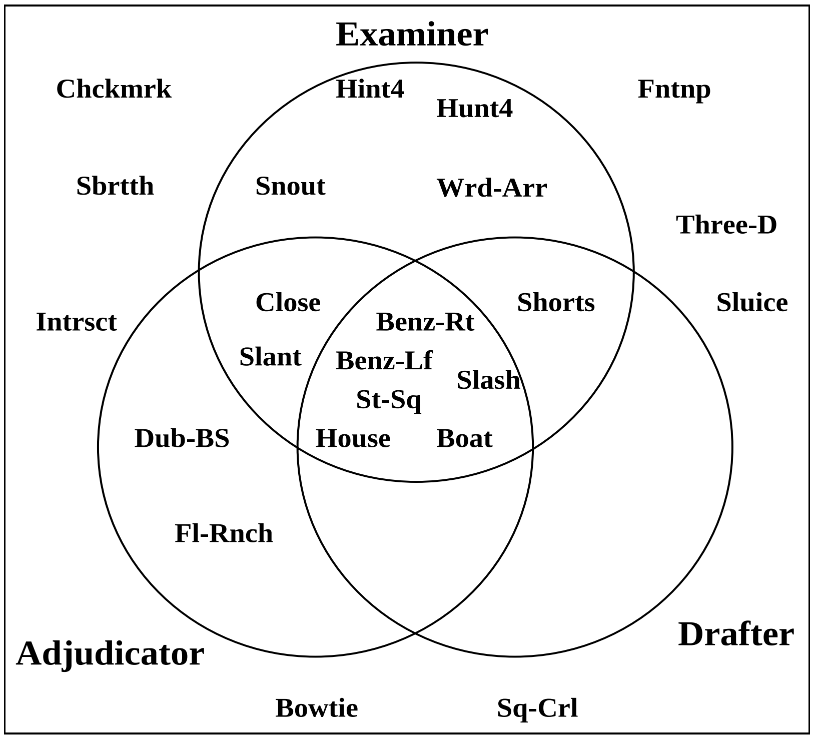

Figure 8.6: Performance by gridfont and by module.

Six of the eight regions in the Venn diagram feature at least one gridfont. The two that stipulate strong Drafter performance without strong Examiner performance are empty, though this was virtually guaranteed by the use of the Letter Spirit rating as a gauge of Drafter performance. Three interesting cases — Double Backslash, Flournoy Ranch, and Shorts — defy the trend of the Examiner–Adjudicator–Drafter cascade.

Table 8.6 lists which example gridfonts fell into which area of the Venn diagram. Each region tells a story. The styles that all three modules can handle well are those that are based upon strong, invariable motifs. In most cases, the definitive motifs

Exam, Adj, Draft	Benz-Lf, Benz-Rt, Boat, House, Slash, St-Sq
Exam, Adj	Close, Slant
Exam, Draft	Shorts
Exam	Hint4, Hunt4, Snout, Weird Arrow
Adj, Draft	*NONE*
Adj	Double Backslash, Flournoy Ranch
Draft	*NONE*
none	Bowtie, Chckmrk, Fntnp, Intrsct, Sabrtth, Sq-Curl, Sluice, Three-D

Table 8.6: Module performance by gridfont.

involve an o-ring motif, though Slash is an exception. Standard Square is also found in this region despite the unusual abstraction (emphasis of letter over style) that defines it. The strength that Letter Spirit modules show in handling squareness has been demonstrated previously, and this effect probably counteracts the trouble that Letter Spirit modules tend to have with abstraction.

Close is handled well by the Examiner and Adjudicator, but not by the Drafter. A possible explanation is that the motif that dominates Close is difficult to incorporate into many letter categories. At first glance this is puzzling, because Slash has a similar motif, and is categorized as being handled well by the Drafter. However, inspection of the actual numerical ratings shows that the two were not rated *very* differently (41.5 and 52.0, respectively); they just happen to be on opposite sides of the arbitrary threshold. The discrepancy that does exist between them may be due to the differences between their motifs. While both motifs key on the diagonal forward-slash across the central zone, Close also calls for the inclusion of an o-ring motif that skirts the right and bottom edges of the central zone. A more elaborate motif (like the one in Close) places additional constraints on drafting (in this case, especially for letters that are not post-and-bowl), and should be expected to make drafting within the style harder to achieve.

Shorts is handled relatively badly by the Adjudicator, but fairly well by the Drafter. One reason for this is the fact that the multiple-choice tests (reported in

Chapter 6) from which the Adjudicator scores were derived happened to place Shorts in test sets with other squarish (and thus, similar) gridfonts, making it difficult for the Adjudicator to distinguish Shorts letters from the non-Shorts distractors, and thereby adversely affecting its score a bit. However, the fact that the Thematic Focus represented Shorts' style (in a word, shortness) as a set of (as far as it could tell) unrelated stylistic properties, each of which carried little importance, may also be a factor. The Drafter can bypass limitations in the powers of the Thematic Focus by employing borrowing, which can imbue some of the gridletters that it produces with a stylistic property that is not necessarily represented in the Thematic Focus — and in fact, six of the gridletters in Shorts actually did inherit their shortness through borrowing. The reason why Shorts is handled well by the Drafter but not by the Adjudicator is probably just the fact that its style can be conveyed better by the Library than by the Thematic Focus.

Hint Four, Hunt Four, Snout, and Checkmark are each handled well only by the Examiner. Consequently, these are gridfonts that are legible (namely, to the Examiner), but that have styles that elude Letter Spirit. The unnatural definition of the gridfonts in the Hint–Hunt family explains why those two fall into this region. The difficulty of representing the style of Snout in the Thematic Focus was discussed in Chapter 6. The interesting contrast with Snout is Shorts. Why does Shorts rank high in Drafter performance, when Snout does not? Most likely, the explanation is squarishness. The Drafter rankings here are based in large part upon Adjudicator scores that were calculated on the Letter Spirit output in the given style. Shorts, being squarish, intrinsically does well in Adjudicator scores; the Adjudicator may not be able to imitate Shorts's shortness, but it sees and likes its squareness. The modules' performance on Snout, which is based upon variable motifs, may be much the same as that on Shorts, but Shorts has an advantage due to its squarishness. Everything said about Snout seems also to apply equally well to the final gridfont in this category, Weird Arrow.

The Adjudicator had solid performance on Double Backslash and Flournoy Ranch despite the fact that they were comparatively difficult for the other two modules to handle.[1] The style of Flournoy Ranch was discussed, in part, in Chapter 6. It has a distinct style that is often in conflict with letter category, and this hurts its legibility. Two hypotheses were offered as candidate explanations of why Flournoy Ranch's style is recognizable. One is that it is expressed in terms of many stylistic properties, no one of which is crucial to the style, and that this kind of style exactly matches the way that the Thematic Focus ends up representing styles. The second is that the weirdness of Flournoy Ranch means that its gridletters are fairly distinctive, and not easily confused with any other style; this is the opposite of the situation with squarish gridfonts, whose gridletters receive good scores but are not easily distinguished from gridletters belonging to other squarish styles. Double Backslash's style is not represented as a set of stylistic properties of lesser importance in the Thematic Focus, but rather, primarily, as a distinct stylistic property for the backslash diagonal that runs across the central zone. Otherwise, Double Backslash's style, though very different from that of Flournoy Ranch in the details, has the same general tendency for its letters to be distinctive in terms of style, but often not extremely strong members of their intended letter categories.

No gridfonts in the set of examples led to good Drafter performance but poor Examiner performance, and so two regions in the Venn diagram are empty. The simple explanation for this is that when the Examiner's performance is poor, the other parts of Letter Spirit cannot function well either. This is why most of the

[1]Obviously, the Examiner had some success with these two gridfonts, or else the Adjudicator would not have had a chance to run on them. They fit into this region because the performance of the Examiner on them is *comparatively* low; these gridfonts both have four or five gridletters that cannot be handled by either module. That is a comparatively bad performance for the Examiner, but not so bad for the Adjudicator.

gridfonts that stymie the Examiner lie in the region around the edge of the Venn diagram, indicating that no module handles them well. The reasons why each of these gridfonts is hard for the Examiner to recognize were discussed already in Chapter 5.

8.3.5 Letter Spirit at its best

As was mentioned back in Chapter 4, the performance of Letter Spirit on those gridfonts that each of its three underlying modules handles well is of particular interest. For these styles, any quirks or shortcomings in overall quality cannot be attributed to a weakness in one module or another. Therefore, the problems in this area must be attributed to limitations in the overall strategy of using modules to carry out review and revision.

Six gridfonts — Benzene Left, Benzene Right, Boat, House, Slash, and Standard Square — fall into this category, the central region of the Venn diagram. Thus, the output for these in the tough promotion condition (seen in Figures 8.3 through 8.5) should be the very best work of which Letter Spirit is capable. Only a handful of the 126 gridletters that Letter Spirit created for these gridfonts fail as solid members of their intended category. As in most Letter Spirit output, the production of good style is a bit weaker, but well over half are successes in this department, too. And some failures, or questionable productions, like the Standard Square ‘z’, are somewhat understandable due to a conflict of letter and spirit.

However, several bad versions of gridletters are seemingly inexplicably bad. For example, why does the Benzene Left ‘d’ cave in, oddly? Why is the Boat ‘d’ taller than the ‘b’? Why does the Boat ‘m’ have a “rounded” upper right corner? What is the matter with the Standard Square ‘m’ and ‘x’? Why does its ‘w’ have an extra quantum curving in? These and perhaps a dozen other gridletters in the test set are markedly flawed. These are the sort of deficits that separate Letter Spirit’s performance from that of a good human gridfont designer. What sort of answer could there be to these questions? Are there dozens of isolated reasons why one case or another is marred? Isolated reasons for many individual flaws in Letter Spirit's work have been identified already; can Letter Spirit's limitations be understood only in terms of an encyclopedic account of each kind of error?

Surely there are many individual combinations of letter and spirit for which Letter Spirit would understandably have trouble producing good output. However, many of the cases that led to trouble in the tough-promotion runs led to impeccable output in the no-promotion and easy-promotion runs. Letter Spirit is capable of designing a good Standard Square 'm', for example. *The essential problem is that it is not guaranteed to.* Letter Spirit operates on the basis of nondeterminism, and this opens the door to erratic behavior. Letter Spirit simply lacks stability from run to run, and the factors that mitigate the nondeterminism, mainly the large number of attempts taken at each category, cannot totally eliminate the erratic nature of its work. Analysis of some output that will be presented below can help explain why this is so.

8.3.6 Style in the face of randomness

It is often instructive to make controlled changes from one test of a program to another. This subsection presents two variations on the kinds of tests done earlier. One set of Letter Spirit runs on SMALL, using BCEFG seeds and tough promotion, began the run with the knob controlling Drafter randomness initially set high (as was the case with a similar test on the Drafter alone, reported in the previous chapter). Figure 8.7 shows the results of these tests. The change in the knob setting leads both to more variety and to more blunders in the output. The first factor — greater variety — increases the likelihood of better gridfonts, while the second — more blunders — decreases it; overall, it appears that the second factor won out. These runs lead to gridfonts that are a bit weaker than those with the regular setting for initial randomness; this can be seen in the moderately higher number of very erratic gridletters in the former case. The ratings from the Examiner and Adjudicator indicate that these gridfonts suffer weaker letter-category strength (by a margin of 20.5 to 15.9) than runs with normal randomization, while style scores are changed very little. One might have hoped that a more random Drafter could, with rare strokes of luck, stumble onto high-quality output that the representation of style built up by the Adjudicator was inadequate to generate. Unfortunately, though, in anything like the current implementation of the program, this hope appears unfounded; high randomness in drafting is too much like blind search within a vast search space.

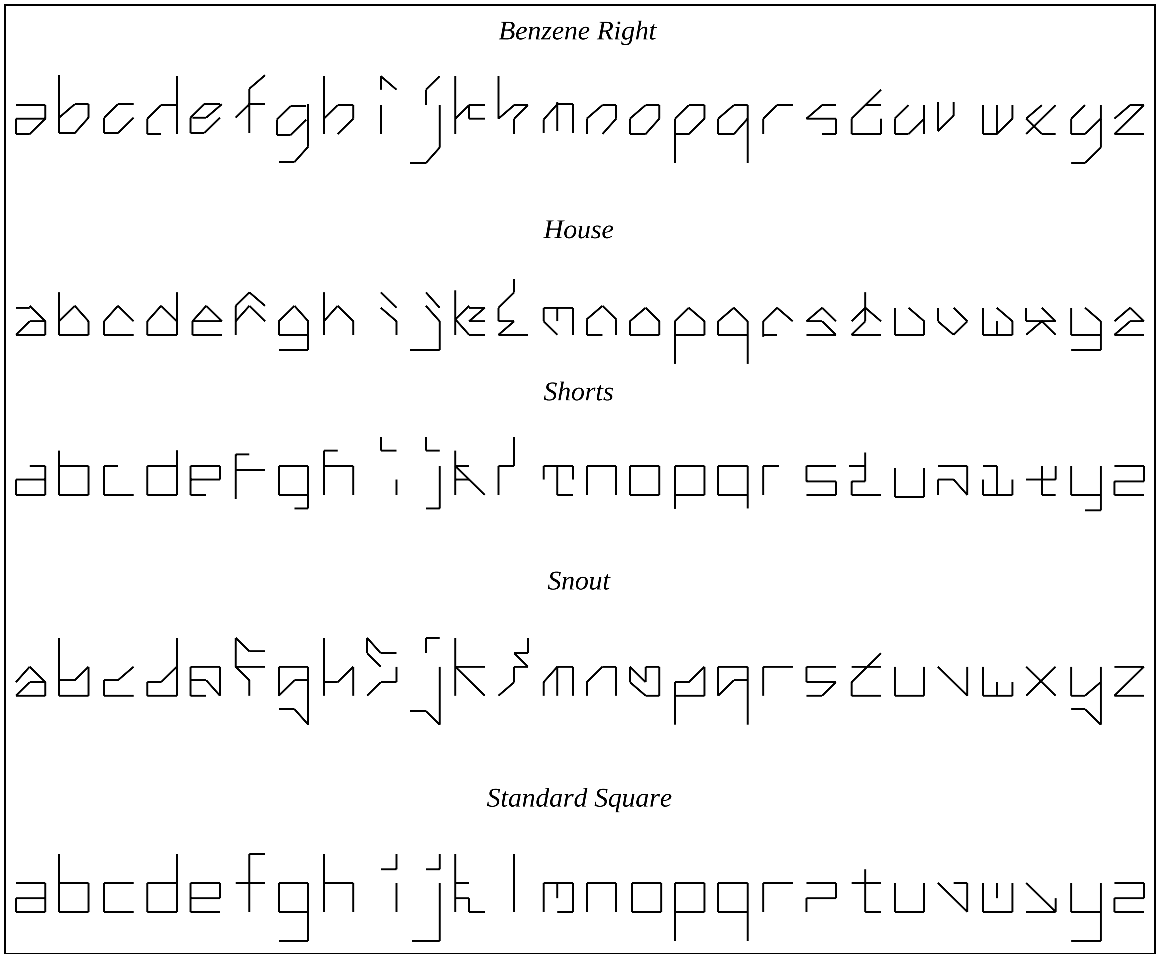

Figure 8.7: Letter Spirit output: High-randomness drafting.

8.3.7 Stability across seed letters

If all tests on Letter Spirit involved only the BCEFG seeds, a skeptic might wonder whether using those particular seeds was essential for the level of quality seen in its output. The extent to which this is true was explored with a test using the SMALL set and tough promotion, but with 'r', 's', 't', 'u', and 'w' ("RSTUW") as the seeds. Figure 8.8 shows these gridfonts; for the first time in this thesis, one may scan the left side of the output and see the program's own output in all of the categories from 'a' to 'g'. The scores from the Examiner and Adjudicator indicate what the viewer senses at a glance — these gridfonts are not as good as those that came from BCEFG seeds. Both the letter and spirit scores suffer by a noticeable margin: 2 to 3 points in both cases. This could raise the suspicion that the program is not robust and that it depends upon particular seeds in order to achieve good results. In fact, this suspicion has merit, but it is true to *some* extent for *any* person or program faced with the Letter Spirit task, since the RSTUW seeds contain less information about most styles than do the BCEFG seeds. A project called *abcdefg* [Adams 1986] investigated ways in which the information in some lowercase letters in a given style could be used to create versions of other lowercase letters in the same style.[2] The letters in BCEFG, and letters like them (such as 'h', which has a post like that of 'b', and 'o', which has the bowl of 'b' and 'g'), are generally the ones that Adams identified as the best sources from which to derive other letters, while RSTUW are among those that needed to be derived from other letters. The dropoff in Letter Spirit performance in this example is thus explained by the fact that the challenge of completing a style based upon the seed letters RSTUW is inherently more difficult than the same challenge with seeds BCEFG.

[2]Superficially, this sounds like the same task that Letter Spirit undertakes; the deep differences are addressed later in this chapter.

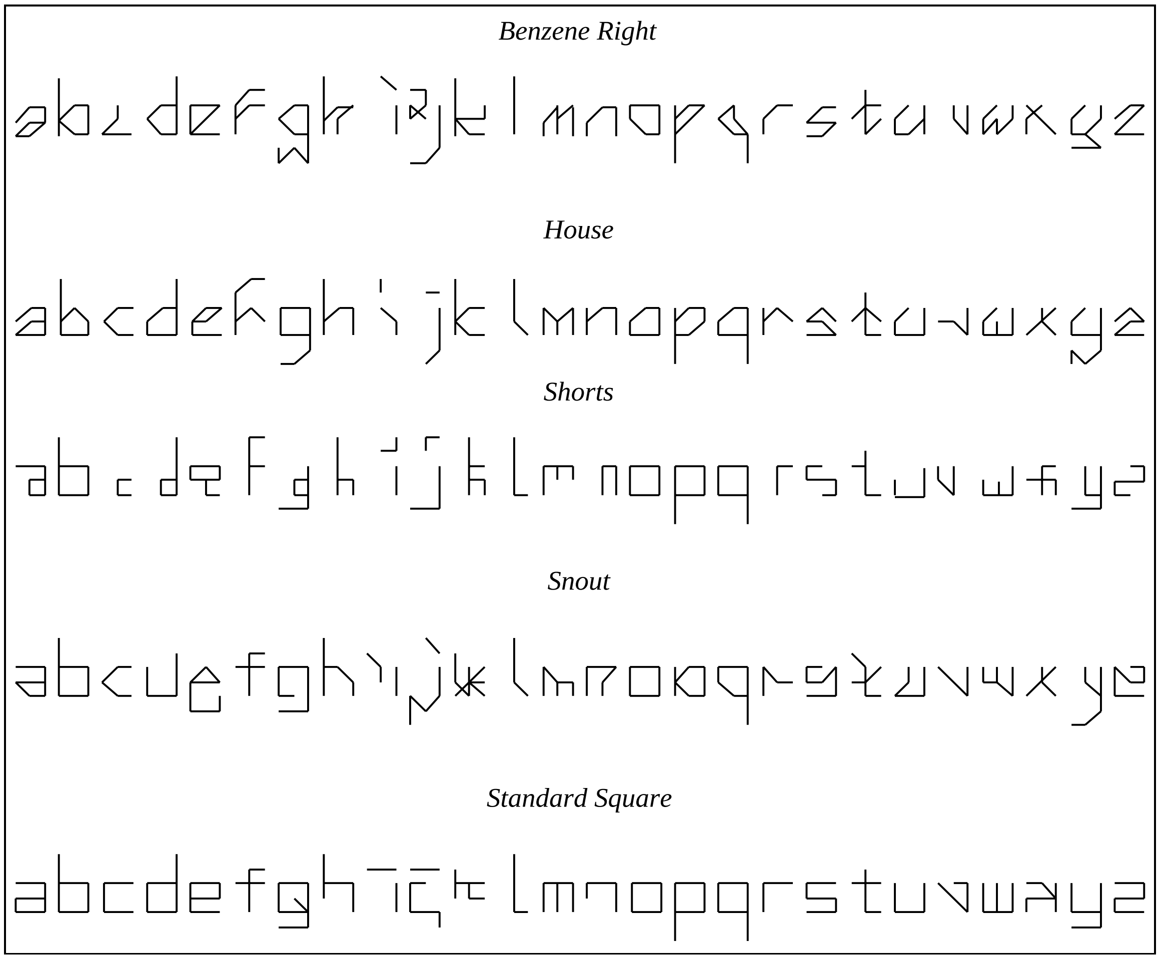

Figure 8.8: Letter Spirit output: RSTUW seeds.

8.3.8 In the long run

In theory, revision strategies promise the ability to provide continuous improvement over the course of a run. One of the most distinguishing characteristics of Letter Spirit is the fact that the quality of a gridfont in progress tends to increase over time. This could lead to the expectation that a long enough run might achieve any desired level of quality. To test this optimistic hypothesis, a test run 3000 drafts long (as opposed to the usual 300) was conducted. This was a no-promotion run, since a run with promotion might end at any earlier point. Because this kind of run is extremely time-consuming, it was carried out only once. Benzene Right was chosen as the style to test, with BCEFG as the training set. The run took about two weeks,[3] and can be contrasted with the 300-draft no-promotion run using the same training set.

Disappointingly, the improvement in quality due to the greater length of the run was negligible, as can be seen in a glance at Figure 8.9. The figure shows the longer run on Benzene Right, preceded by the output of a run with all the same parameters, but allowed to run only a tenth as long; this gridfont appeared earlier in Figure 8.1. For 7 out of 21 non-seed letter categories, the longer run came up with the exact same answers as the short run. In the view of the author, for perhaps 3 of the remaining 14, the short run's answers were better than the long run's, while in 4 cases, the longer run's answers were better. The self-rating by the program, for Examiner score plus Adjudicator score, was nearly identical in the two cases. The longer run was actually rated a bit worse (44.5) than the shorter run (44.4). This is meager payoff indeed for a run ten times as long.

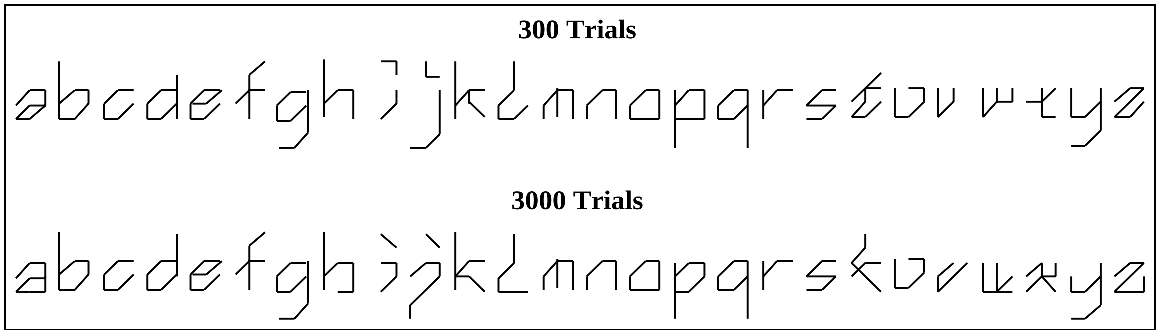

Figure 8.9: Diminishing returns: Similar quality at ten times the cost.

This is not surprising, given the analysis of revision strategies from Chapter 4. While such a strategy is extremely beneficial, the improvement in quality versus trials

[3]This translates to a considerably slower speed than the mean for a shorter run. The reason for this is that competition for memory on a time-shared machine continuously slowed performance, which was initially much faster. Performance is tied to external issues, as well as those of theoretical interest.

is asymptotic, and approaches a ceiling in quality. After several revisions per category, over which a marked improvement takes place, further improvement is hard to come by. Figure 8.10 plots the theoretical increase in quality versus trials from Table 4.2 with that seen over the course of the big run. When appropriate vertical scales are chosen, the predicted increase in quality versus time matches the experimental curve quite closely. (Note that there is no principled reason to match the vertical scales precisely as shown. This simply matches the ranges for the two data sets. However, the similar shapes of the two curves as they approach their asymptotes is clear.) Figure 8.10 actually shows only the first fraction of the big run (575 out of 3000 trials), but subsequent improvement is marginal, with the change in Examiner plus Adjudicator scores amounting to less than 3.5%.

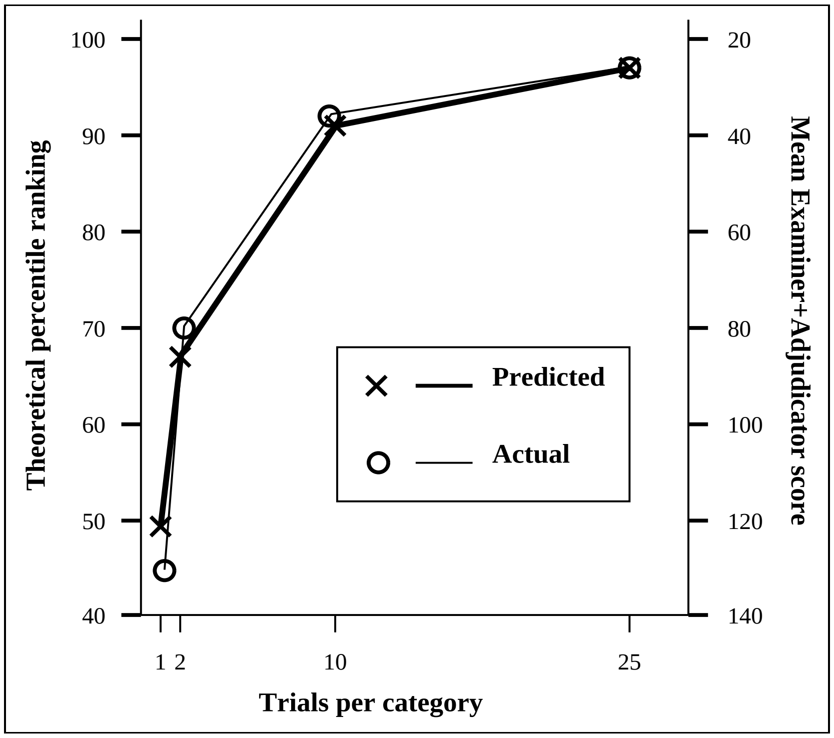

Figure 8.10: Quality vs. trials: theory and practice.

Figure 8.10 expresses the improvement in quality over time. To understand why lengthier runs do not lead to great improvement in quality, it may be useful to consider the same graph with the axes interchanged (Figure 8.11), with quality as the independent variable and time as the dependent variable. This flipped graph exhibits how much time is required to achieve a desired level of quality. The graph climbs quickly; how quickly depends upon the kind of scale used to measure quality. If the scale were open-ended, and if quality were found to grow logarithmically over time, then its alter ego would grow exponentially. If quality has a true ceiling (i.e., a horizontal asymptote in Figure 8.10, which would correspond to a vertical asymptote in Figure 8.11), as the measure based on percentile rank does, then the amount of time required for a given level of quality is infinite for anything at or above the ceiling-level of quality.

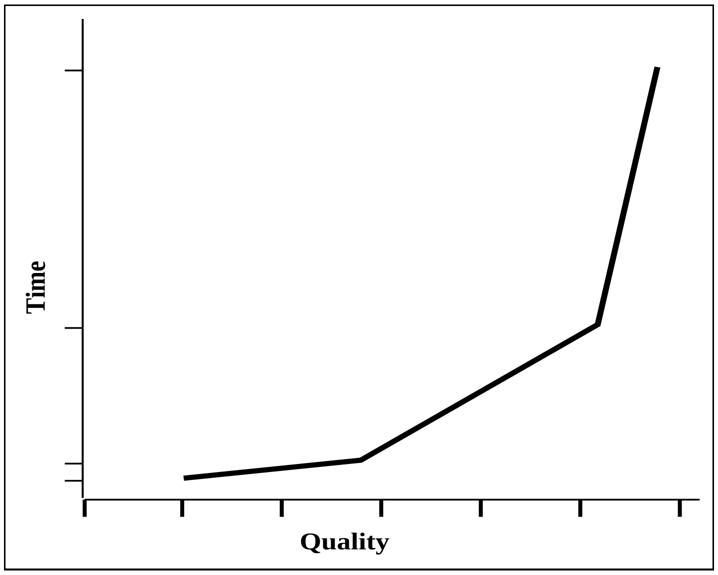

Figure 8.11: The flip side: Time vs. quality.

There is no apparent way, at present, to overcome the fact that the increase in the level of quality of a model of creativity's work levels off after a run has been going on for a while. The way that Letter Spirit fails to keep increasing its output's quality is reminiscent of the way that AM and Eurisko bogged down when they were allowed to run for a long time [Lenat 1982; Lenat 1983]. There is no obvious solution, but in light of Chapter 4's simulation of revision strategies, an intriguing possibility is to decrease the "grain size" upon which the program operates. It was shown that revising a whole gridfont at a time leads to slow improvement, while revising a gridletter at a time leads to faster improvement. Perhaps revising on a lower level, such as that of role-filler, could yield additional benefits. This is something that all human gridfont designers do, and so it is an essential direction for future Letter Spirit research, both for the sake of modeling people more accurately and for the potential boost in performance.

8.3.9 Designing from scratch

Completing a gridfont, given a few seed gridletters, is already a deep cognitive challenge requiring creativity on the part of the designer, whether human or machine. It might seem that an even greater challenge would be that of designing an entire novel gridfont from scratch, with no seeds whatsoever. It is not difficult to have the Letter Spirit program, without any modification, take up such a challenge; all one has to do is start a run in which the set of seeds that is given to it is empty! As was noted in Chapter 7, the Drafter can work just as well with an empty Thematic Focus as it can with a Thematic Focus primed by one or more seeds. So, in the case of runs from scratch, it is simply up to the program (namely, the Drafter) to design its first gridletters without any particular goal style. In runs of this nature, with no seed-evaluation phase, the program begins in the generate-and-evaluate phase. The first gridletter that is generated in such a run may either be accepted by the Examiner, or rejected as a member of its intended letter category. Style ratings are essentially meaningless at this point, because with an empty Thematic Focus, all ratings are very nearly equal.

Whether or not a gridletter is accepted as a tentative member of the gridfont, the run really begins in earnest when a gridletter drawn in the absence of any goal style is promoted to seedhood. At this point, it is used to begin the sense of style, as its stylistic properties are added to the lowest level of the Thematic Focus, and the Library receives the gridletter and its role-fillers, allowing creation of new gridletters based on borrowing.

As was noted in Chapter 5, style is not handled well by Letter Spirit when it is based upon only one gridletter. Letter Spirit was developed with the aim of having it work well when given about five seeds. The methods underlying this proved to generalize well to cases with three or so seeds (assuming the seeds were themselves stylistically coherent) and to larger numbers of seeds (from a dozen or so up to 25), but not with zero to two. Style ratings via the Adjudicator are virtually meaningless (always hovering around a uniform 50) when only one or zero gridletters have been used to build the nascent sense of style in the Thematic Focus. As a consequence, special thresholds for promotion were built into the program for the cases when zero and one gridletters, respectively, have been added to the Thematic Focus; in all other cases, the same formula that works for a handful of seeds works quite well. The problem is that with style ratings relatively invariant while the program seeks the first couple of gridletters upon which to base its style, promotion decisions during that critical time are largely based upon Examiner scores, which do not measure style at all. As a consequence, the struggle for stylistic consistency among the first three gridletters is a great challenge for Letter Spirit in unseeded runs. When this challenge is met, the program does a reasonably good job of creating from scratch — but this requires exceptionally good luck, and it does not usually happen.

Figure 8.12 shows ten gridfonts created by Letter Spirit, working from scratch and using the tough-promotion formula. These gridfonts generally show more consistency *amongst* themselves than *within* themselves. For example, six of the ten ‘o’s are identical, as are six of the ‘x’s.

To show which are the program’s own favorites, Letter Spirit’s ratings of these gridfonts are given in Table 8.7. Number 10 clearly stands out as the best-rated, and visual inspection does show a lot of squareness to this gridfont, as well as frequent truncation, with many role-fillers in some way shorter than they normally would be, particularly the three non-straight descenders of 'g', 'j', and 'y'. In some ways, this gridfont by Letter Spirit is a reinvention of the human-designed gridfont Shorts. The Thematic Focus, at the end of the run, has in the second-highest level (Frequent SPs) abstract rules banning angles of 45° or 135°, and banning line segments longer than three quanta. In the same level are norm violations calling for weight and width to be less than normal, and for many of the dimensions of a role-filler to be pushed inward.

The other nine efforts in this test seems to lack any coherent style, although, as the Examiner ratings bear out, the letter-category strength is generally quite high. In the absence of a coherent style goal, improving letter-category strength is really all the program does during such a run.

	Exam	*Norec*	*Adj*	*Seedified*	*Exam+Adj*
1	9.2	0	46.4	22	55.6
2	8.8	0	36.4	2	45.2
3	13.8	1	49.3	2	63.1
4	9.3	0	49.0	2	58.3
5	27.2	4	47.5	3	74.7
6	15.9	0	46.1	2	62.0
7	12.1	0	47.0	2	59.1
8	8.2	0	51.2	2	59.4
9	5.8	0	47.7	2	53.5
10	12.6	0	11.9	26	24.5
MEAN	12.3	0.5	43.3	6.5	55.6

Table 8.7: Evaluation of self-designed gridfonts: Tough promotion.

Given the difficulty that Letter Spirit has getting started in unseeded runs, this suggests that easy promotion might be the more advantageous strategy in runs of this kind. Figure 8.13 and Table 8.8 put this matter to the test. The number of successes is too small (at most two in either case) to pass judgment on the superiority of one strategy over another. In fact, it seems unlikely that either promotion strategy addresses the critical problem — namely, that building a coherent style from scratch depends upon the rare stroke of good luck that the second and third gridletters that are added to the gridfont happen to have a style similar to that of the first. When this does occur, it does so despite the fact that, as earlier experiments have shown, a Thematic Focus trained solely on one gridletter (in this case, that first gridletter) does not give the Drafter sufficient guidance for it to draft new gridletters in the same style. In any event, the easy-promotion runs show two relative successes. Letter Spirit rates easy-promotion effort Number 7 quite highly. The key to its style appears to be a beveled o-ring motif, with three squared-off corners and one diagonal. This is so common across all the self-designed gridfonts (14 of 20 possible 'o's have this shape) that it may seem unremarkable. However, in this gridfont, many gridletters, including ones that are not post-and-bowl, incorporate a large portion of the motif. The other gridfont in the figure that Letter Spirit recommends highly is Number 9, which, like tough-promotion gridfont Number 10, seems to rediscover some aspects of the Shorts style (though in other ways it is quite unlike Shorts).

	Exam	*Norec*	*Adj*	*Seedified*	*Exam+Adj*
1	10.9	0	42.2	26	53.1
2	8.4	0	51.4	18	59.8
3	19.9	2	48.9	23	68.8
4	10.0	0	48.7	25	58.7
5	12.5	1	49.8	26	62.3
6	9.5	0	45.3	26	54.8
7	12.0	0	14.5	26	26.5
8	8.2	0	50.6	23	58.8
9	19.7	0	9.1	26	28.8
10	11.2	0	52.7	24	63.9
MEAN	12.2	0.3	41.4	24.3	53.6

Table 8.8: Evaluation of self-designed gridfonts: Easy promotion.

Fertile seeds

Earlier, it was noted that not all sets of seeds are equally effective in defining a style; the RSTUW seeds did not lead to output of as high a quality as did the usual BCEFG set.

In Letter Spirit runs that begin from scratch, a poor beginning can keep the program from ever finding a coherent sense of style. This is precisely what happened in 17 of 20 runs. In all of the runs, the program promoted some of its work to seedhood, but the sense of style that was built up in most of those cases was too vague to give rise to a decent gridfont.

In each of the three runs that *did* lead to a fairly coherent style, post-and-bowl letters contributed to the definition of the style very early on. In each of those runs, two of the first three gridletters that were promoted to seedhood, and three of the first five, were gridletters that contained an o-ring motif. In contrast, this was true of only one out of the seventeen runs that failed to find a coherent style. It appears that a good diagnostic of whether or not the program finds a coherent style is whether or not it latches onto two or three post-and-bowl gridletters early on, and promotes them to seedhood. For this to happen requires that the second (and third) letters be similar to the first one, and that, in turn, as was noted earlier, happens relatively rarely.

Style: Evolution and revolution

All Letter Spirit runs that allow promotion allow the program to refine its sense of style throughout a run. This takes place by means of the promotion of individual gridletters. The three runs where Letter Spirit found and implemented a somewhat coherent style while designing from scratch will be used as examples of how style can evolve throughout a run. These runs will be referred to here by their respective strategies and numbers within their test sets: "Tough-10", "Easy-7", and "Easy-9". The way that style can potentially converge can perhaps be considered the top-level program's version of the parallel terraced scan.

A succinct way of describing this process is simply to note when the event of a gridletter being promoted to seedhood occurs. For the three runs discussed here, all twenty-six letters were eventually promoted. The twenty-six acts of promotion were not spaced very evenly throughout the runs; they tended to occur in streaks. In Tough-10, in a streak midway through the run, ten promotions occurred during a span of seventeen attempts — nearly quadruple the number that would be expected if promotions were distributed randomly. Of course, promotions should not be expected to be distributed randomly. This is especially clear in the case of borrowing from the Library, where one promotion can trigger a sequence of promotions that quickly follow causally from the first. Tough-10's streak of ten promotions began with a 'g', which led directly to three other promotions in that streak: 'y', 'j', and 'c'. A streak early in

Easy-7 featured nine promotions within fourteen attempts, as the promotion of one post-and-bowl gridletter triggered the promotion of two other post-and-bowl letters in a span of just four attempts.

Of course, not all streaks of promotion are due to the promotions in the streak triggering one another. One streak in Easy-9 featured six promotions in a span of nine attempts, but not all of these belonged to a self-triggered chain of promotions. The first promotion in the streak was for 'o', but only two of the next five promoted gridletters ('u' and 'c') were rendered by borrowing from that 'o'. The others ('f', 'b', and 'i') just happened by chance to occur at about the same time.

Borrowing is not the only way that style evolution can occur. Subtle changes in the Thematic Focus can lead to style evolution over the course of one or a few promotions. Stepping through an example in detail would entail a rather encyclopedic account (recall the sheer size of the Thematic Focus even at one instant in time). Suffice it to say that stylistic properties do shift among the levels of the Thematic Focus, moving up or down one level, or staying put, with each promotion. Given the size of the Thematic Focus, it is rare for a single promotion to alter it very much. However, the slow drift in goal style that results from individual promotions is clearly a means by which the evolution of style is mediated. Some letter categories (for example, 'l' and 'v') *cannot* be created by borrowing either in part or in whole. In the three examples mentioned above, these categories nonetheless eventually underwent promotion, meaning that the Thematic Focus, as well as the Library, can be the conduit of style evolution. Because streaks of promotion can greatly accelerate the evolution of a gridfont in progress, one may say that changes in style that take place by means of the Thematic Focus demonstrate an *evolution* of style, while changes that take place by means of borrowing from the Library can demonstrate a *revolution* of style. The way that promotions may trigger one another in streaks calls to mind the idea of punctuated equilibria from evolution theory.

8.3.10 A rival approach

Although gridfont design is not the center of a wide body of research, the challenge of completing a gridfont, given some seeds, has been taken up by another group of researchers [Grebert et al. 1992]. In their work, the *GridFont* model was a three-layer connectionist network trained by means of backpropagation. The architecture was relatively simple, and its details need not be reported here. The network was trained on five full gridfonts, plus fourteen gridletters of a sixth style, Hunt Four. It had no knowledge of the letters of the alphabet beyond what it had acquired in its training sessions. Its task was to complete Hunt Four based upon what it had learned from the training sessions.

Because the GridFont network did not have to recognize the letter category of the seeds given to it, the task it faced was a bit simpler than that of Letter Spirit. Also, in the experiment that was reported, the network received *fourteen* seeds so that it could design the remaining twelve letters. It is not clear how robust GridFont's performance would be with a smaller number of seeds; Letter Spirit seems to have a relatively stable performance with as few as three seeds, depending upon which letter categories they are drawn from.

The Letter Spirit program was set the same challenge as that by which GridFont was tested: to design a gridfont, given 14 seeds from Hunt Four. GridFont enjoyed an advantage that Letter Spirit normally does not; that is, GridFont was given each gridletter with the knowledge of what letter category it represented. Letter Spirit received the same benefit for this test, vital because the Hunt Four 'o' is almost always classified as an 'a' by the Examiner. With that gridletter specified to be an 'o', two kinds of Letter Spirit output were generated for comparison with GridFont. The first consists of a single pass by the Drafter at each non-seed letter category, using a Thematic Focus trained on the seeds. The second kind of output is the result of a full Letter Spirit run, with a total of 300 trials over the non-seed letter categories.

The run took place in tough-promotion mode; as it happens, no promotion actually took place. As a final note, obviously the output of the Drafter and Letter Spirit would vary if this task were undertaken more than once; the output here is intended to be representative. Figure 8.14 shows the work of GridFont along with the output generated in the two Letter Spirit tests.

The analysis can begin with the observation that most of GridFont's twelve output gridletters have something seriously wrong with them. In some instances, it seems that editing a quantum or two out of the letterform would help the overall quality of the gridletter tremendously. In others, it seems that something is out of place or missing. The 'z' is completely unrecognizable as such. The strength of GridFont's output is that the o-ring motif of Hunt Four was partially incorporated in most cases. One might charitably guess that the spurious quanta were an attempt to imitate the "twig" on the 'o' that was a seed. The output attracts more attention in the ways that it strays from letter and spirit than in the ways it keeps to them.

The Drafter's output, like that of GridFont, is not perfect. The Drafter makes some questionable moves with the 't' and what is perhaps a spurious quantum of its own on the 'w', while wrecking the 'g' by borrowing the 'j's right-tail and laying it on top of a perfectly fine left-bowl. However, the Drafter's work, with five perfect matches to the human-designed gridfont is, by and large, clearly better than GridFont's. The contrast between the Drafter and GridFont is largely due to their different ways of representing style. The Drafter uses the Thematic Focus and the Library's representation of style; these together seem to capture spirit fairly well, and in a way that can be integrated with the Conceptual Memory's role-based representation of letter. Letter Spirit's role-based approach tends to respect structural properties of letterforms, such as their expected topology. GridFont has a distributed representation of letter and spirit, though it is not necessarily indicative of what a more elaborate connectionist architecture might be like. GridFont clearly has some ability to produce output that reflects both letter and spirit, although not very consistently, and seldom are both letter and spirit strong in the same letterform.

Letter Spirit as a whole is like the Drafter in terms of the representations that it uses to create output, but is unlike both GridFont and the Drafter in that it uses a revision strategy. The three-way comparison makes the power of revision obvious. Letter Spirit's output matches that of the human designer in seven of twelve instances, and, aside from a quirky jog in the 'y's tail, has no weak output whatsoever. It manages to miss the exact 'i' from the human-designed gridfont that both GridFont and the Drafter chose, but otherwise it matches and exceeds the work of the other two approaches. A comparison between Letter Spirit and GridFont that emphasized only those ways that the Drafter and GridFont are different would miss most of the point of Letter Spirit. The conceptual representations of letter and spirit and their cognitive plausibility are important, but the revision strategy, its cognitive plausibility, and its power are crucial as well.

8.3.11 Other programs involved with typeface creation

The Daffodil program takes as input a small set of decorative strokes and flourishes provided by the user, and then makes a blind, straightforward substitution of those strokes into renditions of each of the letters [Nanard et al. 1989]. The *abcdefg* model (described in detail, but only partially implemented) employed a very similar technique, but takes as its input a small number of "control characters," provided by the user, which are then cannibalized for parts which can be reassembled (this step also requires user input to specify the kinds of serifs and soft curves that are needed to connect the parts) into the remaining letters of the typeface [Adams 1986]. Finally, the Metafont program uses general descriptions of letters that a user can manipulate, by setting the values of a large number of parameters, into a wide range of typefaces [Knuth 1982]. All three of these programs are better classified as design tools than models of human creativity. In essence, they have the same relation to creating typefaces as a compass has to drawing a circle. While they aid in the production of a fine finished product, the work is really done by the hand of the human wielding the tool.

8.4 Conclusion

This chapter has provided a description of the program's top-level control and has given many examples of its output, along with basic explanations for why Letter Spirit did some of the things it did. Each of the previous three chapters described one module of Letter Spirit and ended with a summary of that module's strengths and shortcomings. For the Letter Spirit program as a whole, a proper summary must go far beyond that of any of its component modules and — in fact — must discuss them, as well, to be complete. Consequently, the final comments on Letter Spirit serve as the basis for the chapter to follow.

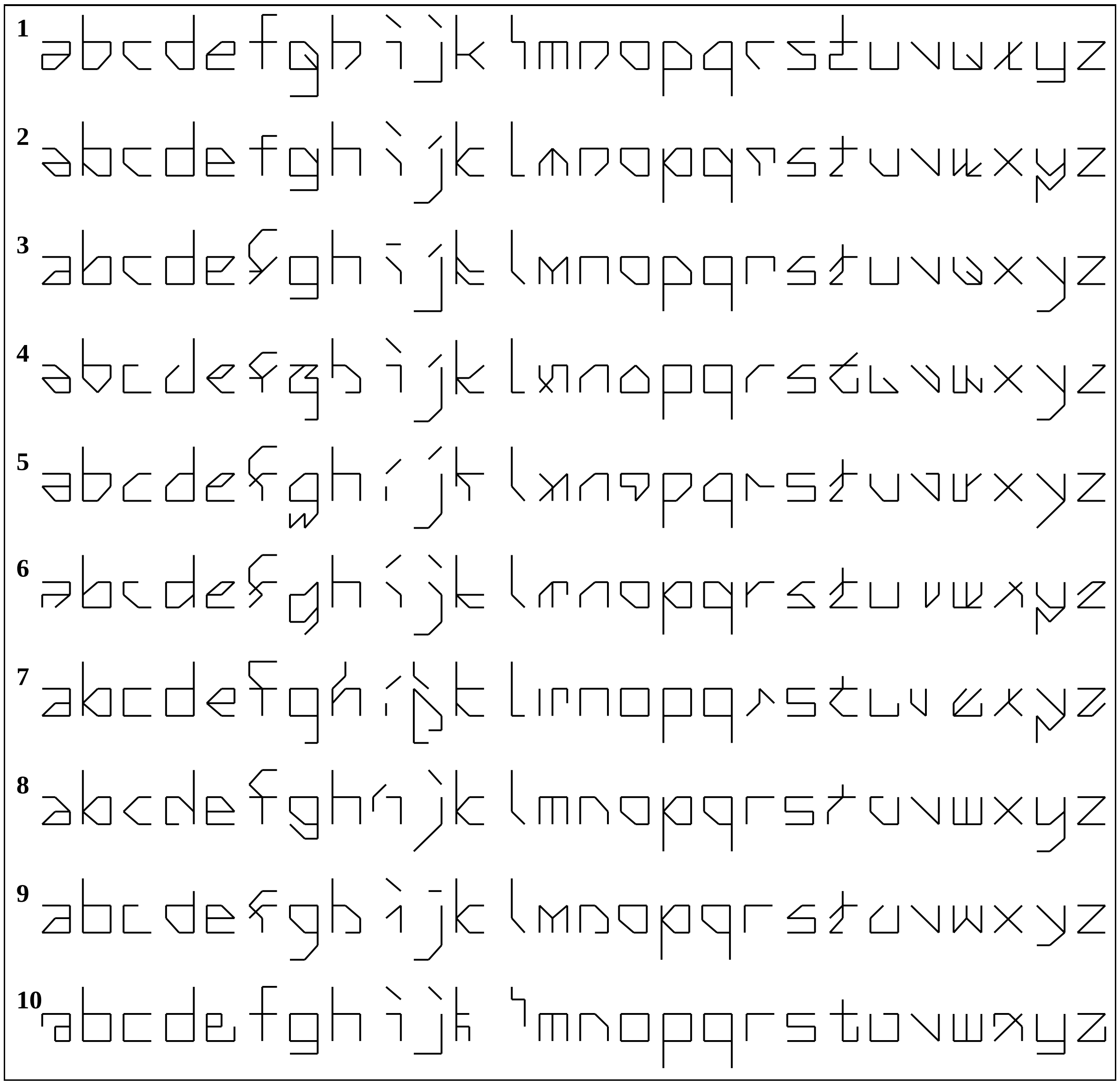

Figure 8.12: Letter Spirit output: No seeds, tough promotion.

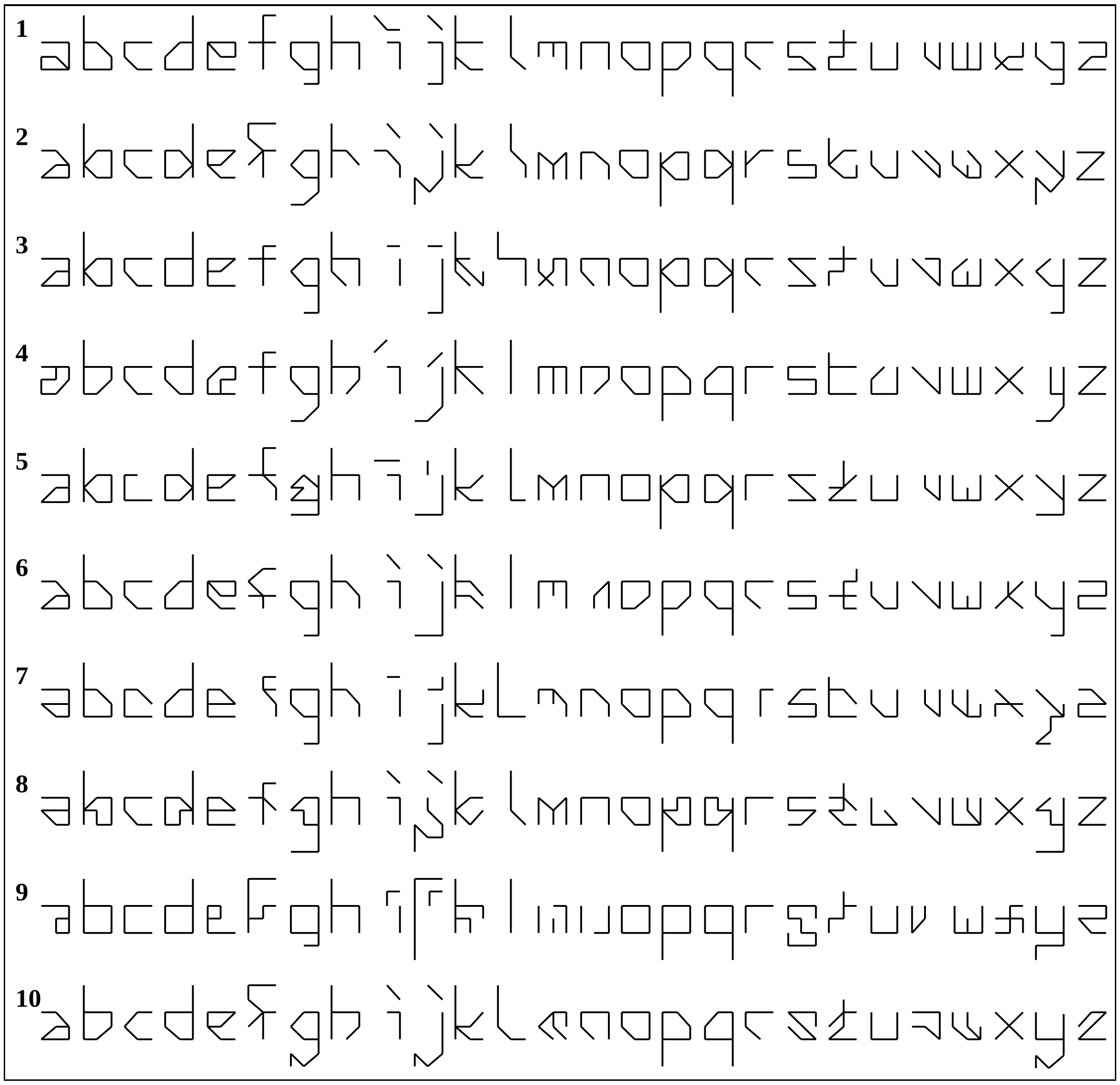

Figure 8.13: Letter Spirit output: No seeds, easy promotion.

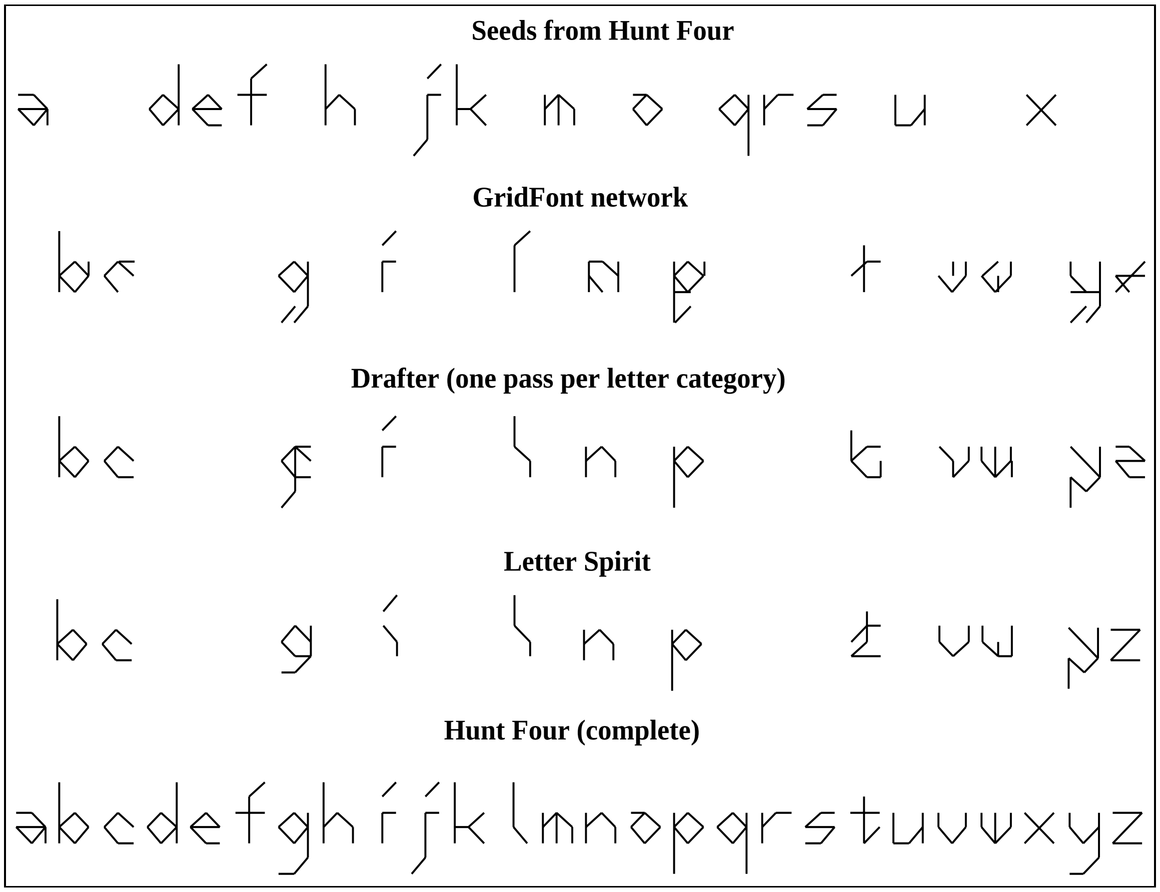

Figure 8.14: Three approaches to the Letter Spirit challenge.

CHAPTER NINE

Strengths and Shortcomings

9.1 Introduction

The goal for the Letter Spirit program, as it had been stated before implementation had progressed very far, was that "starting with one or more seed letters representing the beginnings of a style, the program will attempt to create the rest of the alphabet in such a way that all 26 letters share that same style, or *spirit*" [Hofstadter and FARG 1995]. The work reported in this thesis should be called a qualified success, since the program's output meets this goal in some instances, falls short of it in others, and exceeds it in a few others (most notably in its ability to generate coherent gridfonts without having any seed letters given to it). However, the program's shortcomings can actually contribute positively to how well the Letter Spirit *project* succeeds in investigating the processes underlying human creativity. The Letter Spirit program contributes to this investigation not only in the ways that it succeeds in modeling human creativity but also in the ways that it contrasts with it by behaving in a manner that is inferior to (or simply different) from it.

The discussion in this chapter addresses not only the behavior of the full Letter Spirit program, but also that of its individual modules. The character of the Letter Spirit program as a whole is shaped by the details of each of the four parts of the program (the Examiner, the Adjudicator, the Drafter, and the top-level control); the goal of this chapter is to provide a concise and complete analysis of Letter Spirit that draws upon the observations laid out in Chapters 5 through 8.

The following sections of this chapter discuss first the strengths, then the shortcomings, of the Letter Spirit program. This discussion is broken down by considering the different parts of the program in turn, and how well, with regard to any particular portion of the program, the following three worthy goals are met:

1. Cognitive plausibility.
2. Level of quality of the output.
3. Efficiency with which the module/program runs.

That organization will be applied to the upcoming sections of this chapter.

9.1.1 Strengths

Examiner

Of all the parts of the Letter Spirit program, the Examiner is the one whose task makes it easiest to compare program and human performance. Such comparisons have been performed, and they show that the Examiner excels at letter recognition in many ways. This is evident in the rate of its correct responses: the 1999 version of the Examiner outperformed experimental subjects 93.5% to 84.0% in correct responses (Table 5.3).

While the Examiner is perhaps more accurate than people, it is nonetheless a good model of letter recognition; it is almost certainly the Letter Spirit module that most accurately models human behavior. McGraw [1995] showed that the 1995 Examiner's performance (in terms of error-making) matched that of people quite closely. All versions of the Examiner (more so in each version than in the one that came before it) incorporate a number of features that are apt for modeling human letter recognition as well as human cognition in general. Most crucial among these features are the following:

1. A quick-look gestalt function is used as a heuristic to guide subsequent processing.

2. Interaction between top-down and bottom-up processing is mediated by spreading activation.

3. Letters are represented as the composition of roles.

4. Categorization is carried out by the tentative building-up and taking-apart of symbolic structures that capture, among other things, the level of roles and the level of letters.

McGraw's analysis also showed that the output of the 1995 Examiner matched that of people much more closely than did several other letter-recognizing programs that were based upon other approaches to the task. The shortcomings of the particular programs that were tested does not indicate that any broad classes of approach are inherently flawed, but it does suggest that the approach taken by Letter Spirit is more cognitively plausible than some other reasonable approaches. In addition, one should note the case wherein one of the models that McGraw tested, a three-layer connectionist network, achieved humanlike raw performance in terms of number of correct identifications, but made errors that were very different from the errors a person would make — some errors that no person ever made in a set of experiments. Consequently, the Examiner's behavior correlated with human performance much better than did that of the connectionist model. This case points out that a high rate of correct categorizations does not necessarily indicate humanlike behavior, while the Examiner, in comparison, offers both.

Beyond the fact that the Examiner has a fair measure of cognitive plausibility while achieving a high rate of correct recognition, it also excels as an engineering solution to the problem of letter recognition. With the 1999 overhaul, it is markedly more accurate than the people tested on the same task. The optimizations made to the Examiner show the value of opportunistic design — the making use of every piece of information the program has that might bear on subsequent decisions. The earlier version of the program, which had a network that stored activations for each role and role-set concept, changed the activations throughout a run in wild, unprincipled ways so that activation *generally* described the relevance of a concept to the current run, but could not be counted upon to state precisely *how relevant.* With good reason, concept activations were not used as the basis for many decisions within the old Examiner. In the optimization, the effort was made to make activations as meaningful as possible, so that if one concept was more active than another, that indicated a serious likelihood that the former was more relevant to the run than the latter. When the program was so changed, it was found that activations could be used as an informative factor in almost every step of processing involving roles and role-sets. In effect, this means that every process that sets activations is an investment that pays off at numerous subsequent points in time when the program makes more-informed decisions because it factors in how active concepts are. This overall strategy of having activations be meaningful and bringing that information to bear later serves as a powerful optimization in terms of computational efficiency — one that might be used even in systems that are not intended to model cognition.

Adjudicator

The Adjudicator builds representations of visual style, a phenomenon that has long defied any detailed understanding. Unlike the Examiner, which has a task like that of commercial systems for optical character recognition or handwriting recognition, the Adjudicator operates in a domain that is uniquely its own. Accordingly, there are no points of comparison for its dual tasks of first building up a representation of gridfont style and, second, making use of that representation for style recognition. The representation is also used by the Drafter to create according to the given style, so a complete test of the Adjudicator's performance would have to look at Drafter output. However, the quality of Drafter output is due not only to the work done by the Adjudicator, but also, naturally, the work done by the Drafter itself. The dependency of Letter Spirit modules upon one another thus makes it difficult to gauge the quality of the performance of the Adjudicator or Drafter (or Letter Spirit's top-level loop) in as thorough manner as was done with the Examiner. However, a few observations can still be made, as follows.

Several characteristics of the Adjudicator capture aspects of human cognition in the given domain. For one, the nondeterministic Coderack basis of the FARG architecture is employed, and it has already been argued (in this thesis and elsewhere) why that is a boon to modeling human cognition.

One way in which the Adjudicator clearly achieves a certain level of cognitive plausibility comes from the fact that the stylistic properties built into it include many of the more basic ones that are useful in describing fonts' styles,

The representations that the Adjudicator creates (and stores in the Thematic Focus) are structured according to the frequency with which stylistic properties occur in the gridfont being worked on. More salient ones are placed in high levels and have a high degree of influence on subsequent behavior. Less salient ones, usually far more numerous, have just a small degree of influence, particularly if they rise in frequency as more letters are added to the gridfont over time. As was argued in Chapter 6, this way of representing style is nicely consonant with the truism (expressed at length in Chapter 3) that people have difficulty articulating the basis for any particular style. In terms of the Thematic Focus, this is explained by only the most important stylistic properties being accessible for conscious attention; those stylistic properties make up an important, but incomplete, description of the style. Thus, in making stylistic judgments, while people can (unconsciously) make use of a wide spectrum of stylistic properties, including those that would be found in the Thematic Focus's lower levels, they can report only on a subset of the style's makeup, even as they utilize the more richly textured description of style that includes less frequent and less central stylistic properties such as are found on the lower levels of the Thematic Focus.

Stylistic properties have their influence moderated not only by their frequency, but also by a metric of salience, which, like frequency, is also used to weight how much the presence or nonpresence of a given stylistic property is used in calculations of how well an input gridletter matches the style in the Thematic Focus. These metrics are intended to reflect measures of salience that are implicit in human stylistic judgments, although the degree to which they make the Adjudicator's behavior more humanlike is hard to estimate precisely.

In terms of performance, the Adjudicator scores quite respectably for that of a first effort in its domain. Posed with a multiple-choice test in which it must select the gridletter that best fits a given style (thus, when its choices are implicitly compared with the judgment of the gridfont designer), it achieves about 60% accuracy over data sets that include some moderately complex and subtle styles; with data sets composed of more basic styles, it can achieve 85% or higher.

Discussion of the Adjudicator's efficiency is quite straightforward — literally. The Adjudicator runs like a straight-line program, whereas the Examiner and other FARG programs have implicit loop-like behavior, with new codelets posted liberally throughout a run, as a great deal of building-up and tearing-down of structural representations is necessitated by the search for a final answer. Lacking this, a single run of the Adjudicator merely seeks out stylistic properties in the input gridletter and creates bridges between those and stylistic properties in the Thematic Focus, and the amount of effort devoted to those high-level goals is invariant from run to run; this leaves little room for improvement in terms of efficiency within runs.

The one important idea relating to efficiency in the implementation of the Adjudicator concerns the way in which bridges between stylistic properties in the Thematic Focus (the representation of the gridfont's style) and ones in the Workspace (the representation of a single gridletter's style) are created by picking a stylistic property on one side, nondeterministically, and looking for possible matches on the other side. It is necessary to perform this operation in the Workspace-to-Thematic Focus direction, so that stylistic properties that are in the input may be noticed for the first time during this Letter Spirit run. The added efficiency over the course of several runs comes from also checking in the Thematic Focus-to-Workspace direction, with a greater emphasis on the higher levels of the Thematic Focus, so that fewer random looks are required in order to ensure that the most important (that is, most frequently-occurring in the gridfont) stylistic properties are checked for their presence or nonpresence in almost every Adjudicator run.

Overall, it is difficult to evaluate the Adjudicator in isolation because so much of its purpose is to assist the Drafter (and by extension, the Letter Spirit program as a whole) in the creation of gridletters (and by extension, gridfonts). While it is hard to say precisely *how* well the Adjudicator performs its task, it is apparent that it enjoys some measure of success in terms of perceiving, representing, and recognizing style.

Drafter

The Drafter models, to some degree of accuracy, methods of rendering gridletters that can clearly be seen in human gridfont (and typeface) design. It produces gridletters by any of three different methods, all three being strategies that humans employ in designing gridletters. Two of those strategies involve borrowing shapes from other gridletters that are already part of the gridfont in progress, while the third, Coderack drafting, models the way that a gridletter can be rendered point to point, as though it were being created by a writing utensil drawing on a physical medium. It is beyond dispute that all three methods are broadly like ones that humans employ, although it is also clear that the Drafter's implementation falls far short of capturing all of the details concerning how people carry out those methods. In fact, the Drafter's implementation of the methods that borrow shapes from other gridletters hardly *has* any details underlying it; those processes are implemented as very straightforward computer operations.

Coderack drafting, on the other hand, is meant to capture the way that multiple pressures can collectively influence a decision — namely, which quantum should be drawn next. The manner in which the pressures interact to produce a drafting decision is intended to be a good approximation of how the same kinds of pressures interact in a person choosing an individual quantum while drawing a gridletter, even though the person certainly has a richer set of pressures influencing their decision than does the Drafter. Specifically, the process is meant to model how multiple pressures (many of them unconscious) produce a conscious decision that leads, in turn, to a motor response.

Drafter output differs critically from human gridletters, in part because the Drafter does not evaluate its output very much before committing to it; this is understandable since the Letter Spirit architecture calls upon the other modules to perform that evaluation. Human output that has not been evaluated before it was presented to the world is difficult if not impossible to come by, so it is hard to make a fair assessment of the Drafter's output by comparing it to that of people.

In terms of efficiency, the Drafter, during Coderack drafting, speeds its evaluation of candidates for the next quantum by eliminating candidates from consideration once their rating has fallen sufficiently far behind the candidate that is rated best. This has little effect on the actual output (because any candidate far enough behind to be pruned had almost no chance of being chosen anyway), but does improve the program's speed and is an application of the parallel terraced scan.

The Letter Spirit loop

Letter Spirit's top-level program carries out a fairly simple scheme for coordinating the three complex modules. The top-level loop's activity is meant to mirror that of people involved in the same task, and it captures a few key aspects of human creative behavior.

Chief among the humanlike characteristics of Letter Spirit is its review-and-revision strategy, whereby the Examiner and Adjudicator are utilized to evaluate the output of the Drafter; the top-level program keeps only those gridletters that receive sufficiently good ratings. Rather than quitting when a single attempt at each letter has been produced, the program continues on long afterwards, producing unmistakable improvement in the quality of its work on a gridfont as a run progresses. The gridletters that it produces early in a run and that human observers agree are flawed are the very ones that it targets for replacement via more successive attempts at rendering; usually, better replacements are found. This continues until the rate of improvement eventually decreases and the run comes to an end. In these ways, the Letter Spirit loop, "the central feedback loop of creativity", mimics the activity that is evident in human creative endeavors.

In addition, in some modes that were tested, Letter Spirit is capable of updating, during any given run, the sense of style guiding its creation and evaluation of new gridletters. This too is an important element of human creative behavior.

Also, Letter Spirit is capable of creating full gridfonts that are stylistically consistent (some of the time) even when it is given no seeds. This kind of creating “from scratch” strikes some people as a more authentic kind of creativity than Letter Spirit’s finishing a gridfont that has been started for it.

Another appealing feature of the loop is that it allows one module sometimes to make up for the shortcomings of another. One major goal of Letter Spirit’s design was that the Examiner and the Adjudicator should censor the Drafter’s worst attempts. That goal has been realized. However, the opposite is also true; sometimes when the Examiner and Adjudicator would allow inferior gridletters into the final gridfont, the Drafter manages to produce output that satisfies the intended style anyway and thereby covers the shortcomings of the other two modules just as they, in other cases, cover the Drafter’s shortcomings. An example of this can be seen in the creation of Shorts, when the Adjudicator would be fairly happy with letters that were squarish but not particularly short anywhere, but the Drafter nonetheless tends to produce letters that are short in places.

Crucial to the success of Letter Spirit, the parallel terraced scan is an integral feature of all FARG models, and it can be seen to assist in cognitive plausibility and efficiency in all three Letter Spirit modules as well as in the top-level program. Its benefits essentially come from the fact that, by utilizing information that is accrued during a run, it enables certain decisions to be made by the program at runtime rather than by the programmer at programming time, which is the central goal of artificial intelligence. The parallel terraced scan is an instance of AI operating on a meta-level, with a program making decisions (hopefully, intelligently) about its own processing.

All three of the modules and the Letter Spirit loop incorporate, each in its own way, temperature — a means of modulating the randomness of behavior — or something very much like it. The Letter Spirit loop, for instance, decreases the amount that style influences the Drafter's attempts at rendering a particular letter category when it has already undergone several attempts at drafting without the Examiner having recognized any of them as members of the intended letter category. It also turns up the amount of randomness in drafting as a run goes on, under the premise that novel letterforms are more useful at that point than good ones that had been drafted previously. In all of its guises, the use of temperature employs the principle encapsulated in the phrase "desperate times call for desperate measures" — a principle seen at work in many kinds of human behavior.

Nondeterminism is another crucial characteristic of Letter Spirit. As was noted earlier, review-and-revision strategies make sense only if successive attempts at the same task lead to different output; if every attempt is going to be identical, then there is no point in generating more than one. It is thus the Drafter's nondeterministic approach that makes the Letter Spirit loop work. Nondeterminism is also helpful to the Examiner, which often makes more than one attempt at its own task (parsing the input); again, nondeterminism is needed to make this approach worthwhile. It is believed (although it has not been proved) that nondeterminism is also beneficial to the Adjudicator's work.

Finally, the entire structure of Letter Spirit is intended to capture a spectrum of nested behaviors ranging from higher levels to lower levels in ways that do justice to the differences in human activities across those levels. The codelet-level behavior of each of the three modules is intended to capture nuances of unconscious behavior, while the higher-level behavior of each module, and certainly that of the Letter Spirit loop, are meant to model conscious behavior. The framework of distributed and algorithmic processes was introduced in Chapter 2 to describe some aspects of this distinction that is so important to models that try to encompass behavior spanning just such a variety of levels.

In summary, Letter Spirit creates gridfonts while modeling closely many aspects of the behavior of a person engaged in the same task. This emerges in part from the general match between the activities supporting human creativity and the relatively simple top-level program of Letter Spirit, depending crucially upon the sophistication of the three underlying modules.

9.1.2 Shortcomings

We will not make the same old mistakes. We will make our own.

— Henry Kissinger

Despite the strengths of Letter Spirit, it is currently a very long way from capturing all the subtlety of human creativity in the gridfont domain; it is unlikely that subsequent work amounting even to several times the amount of work that has gone into the project so far would "finish" Letter Spirit by making it completely human-like with regard to its task. The work described in this thesis was undertaken with the goal of deriving as much benefit as possible from understanding the weaknesses, as well as the strengths, of the program, and this section attempts to comprehensively list the most important shortcomings of the program, most of which have been mentioned in earlier chapters.

Representation of letters and roles

Roles and role-sets are hard-wired into the Conceptual Memory, which does not allow the dynamic creation of new ways of parsing letters of a given letter category in response to an unusual input.

When the program recognizes a gridletter by means of one of its letter conceptualizations, it cannot later see that gridletter according to one of its other letter conceptualizations and use both as inspirations for other gridletters. For example, an ideal gridfont designer could hypothetically see a ‘b’ as a ‘circle’ role-filler and a two-quantum ‘left-post’ role-filler and use the ‘circle’ in the subsequent design of an ‘o’. Later, the person could decide to try to re-parse the same ‘b’ as a ‘right-bowl’ and a long four-quantum ‘left-post’ and use the ‘left-post’ in the subsequent design on an ‘h’. Letter Spirit currently lacks that kind of flexibility.

There are many possible r-roles that would enrich the representation of letters; examples include the angle at which role-fillers should meet and the desirable distance between role-fillers that do not meet.

The Examiner has several built-in limitations in the way it can segment a gridletter into parts; this entails limitations on how it can parse a gridletter into roles. It does not allow segmentations that feature discontinuous parts or parts that have three tips, or in which two parts share a quantum or divide one quantum between them.

The Examiner also lacks the ability to work with negative space — to see a role as being filled by the empty space that is bound by quanta. A similar limitation is that it cannot see a tip (like the up-arm of a ‘k’) in a 45° angle indicated by a kink in an appropriately-oriented part. Another related (though rarely relevant) problem is that it cannot see three-dimensional shapes that are implied by two-dimensional configurations of quanta on the grid.

Representation of style

The set of abstract rules that Letter Spirit can work with is completely fixed, having been hard-wired into the programming. This is largely true for norm violations as well, although there is a small amount of productivity in that relative norm violations can be registered for any possible combination of dimension values (for example, “short→tall”), appropriate to what is found in the actual gridletters. Motifs are highly productive, but their implementation could benefit from being *more* hard-wired. Namely, motifs that seem to people to be more basic structures, such as squares or long collinear sequences of quanta, should be more easily noticed than relatively haphazard combinations of quanta. In summary, each type of stylistic property (SP) should be implemented both through some hard-wired information and through some general means of productivity; currently, none of the three has been implemented with the proper sophistication.

While the Conceptual Memory has nodes representing role and role-set concepts (a feature of the program that is utilized by the Examiner), it has no nodes for style concepts that the Adjudicator and Drafter might use to signify when one or another stylistic property is active; Letter Spirit could probably benefit from being extended in that way. The implementation of a spreading-activation network handling style concepts would ideally take place by means of something more like a Slipnet like that in Copycat, rather than a simple localist connectionist network such as the one used by the Examiner. A Workspace that could form novel descriptions of style based on a style-Slipnet would be helpful for the goal of bringing productivity to all SP types.

A subtle but serious shortcoming in Letter Spirit is that it is not flexible in specifying how often a given stylistic property should appear. A human-concocted style might call for a given stylistic property to appear once per role-filler, once per gridletter, twice per gridletter, on a quantum-by-quantum basis, or quantified in any of several similar ways. Letter Spirit represents some kinds of style on a role-level (in the case of norm violations) or even the quantum-level (motifs), but cannot flexibly moderate its notion of how often a given stylistic property should appear depending upon the situation.

The set of stylistic properties in Letter Spirit is quite limited. Ideally, all three kinds of SP would be defined in ways that allowed the program to abstract from them and thereby to generate an endless variety of new composite SPs. The ability to generate novel SPs should include the ability to notice environmental factors related to SPs, like context, salience, and flow (explained at length in Chapter 1).

Letter Spirit's way of handling styles as a conjunction of stylistic properties has limitations, as is shown by the problems that Letter Spirit has with gridfonts like Shorts and Flournoy Ranch. People can conceive of styles that are conjunctions, disjunctions, or negations of stylistic properties, or that are formed by composition of logical operations; moreover people seem to be able to conceive of styles that combine their underlying stylistic properties in ways that are richer and more flexible than what formal boolean logic captures adequately. Many issues in how people represent categories have yet to be understood before a proper solution can be suggested, much less implemented.

The relative importance of style and letter-category strength are not weighed against each other in a purposeful way; for example, in the mind of the designer of Standard Square, style was meant to be weak, so that letter-category strength would prevail whenever it and Standard Square's style (squarishness) conflicted. People sometimes deliberately slip letter in favor of style or vice versa, but Letter Spirit always explicitly tries to make both as strong as possible in every gridletter (although it may fail to do so in particular cases).

R-roles, specifying how role-fillers are meant to touch one another are not adequately implemented. The ways that things can touch one another and how the norms for touching can be slipped greatly exceed the program's ability to handle. Although that sort of variety does not come to play in many styles, the way that this is manipulated in some human-designed styles suggests tremendous potential variety.

In general, people have the ability to abstract style in ways that are much more flexible than Letter Spirit's ability to do so. People can continually step outside the perceived limitations of a domain to create in new ways, while Letter Spirit, for the most part, combines a finite set of primitive stylistic properties into a set of styles that it can represent that is vast, but that has boundaries that seem to the human observer to be distinctly limited. The set of 23 sample gridfonts includes many that illustrate styles that Letter Spirit does not capture very well (or, in some cases, at all).

Examiner processing

The Examiner suffers tunnel vision in that it halts whenever it first finds what seems to be an acceptable answer, while a person might find multiple answers but indicate the one that is best. People seem to be instantaneously aware of several possibilities in certain ambiguous letterforms, whereas the Examiner lunges for the first answer it can find and then stops. If future improvements are made to the Examiner, an approach incorporating limited parallelism might be the best solution, wherein a small number of what appear to be the most promising possible answers are all pursued, with an attempt to segment and then parse the gridletter as each of those, after which the final answer would be chosen from whichever of them is rated best.

Adjudicator processing

Adjudicator scores, at present, are highly dependent upon context; only with the style and the letter category of the input held fixed do the scores have much meaning. A score of 25 could be excellent for one style's rating of 'z's, but terrible for another style's ratings of 'o's. This makes it very hard to decide when a gridletter should be promoted to seedhood.

The Adjudicator does not note how motifs are situated within a gridletter in terms of context, salience, and flow.

The Adjudicator does not currently look for stylistic consistency *within* a gridletter — something which is just as much a positive indication of adherence to a style as is across-letter consistency. For example, when the four tips of an 'x' all have a similar shape at the end, or when a ‘v’ is symmetrical, that indicates that the shapes indicated by the ‘x’s and ‘v’s tips are being reinforced. Letter Spirit cannot see things of this nature as repeated manifestations of the same stylistic property. If the features in question strain letter-category strength, Letter Spirit ends up regarding the repeated instances of the feature merely as separate quirks detrimental to letter without noting that they are a positive statement of that feature as a part of the style.

Many stylistic properties could not possibly be incorporated within certain letter categories, or could do so only if the letter-category strength were severely compromised. For example, it is impossible to enforce the ‘ban-diagonals’ abstract rule in ‘v’s and difficult to do so in ‘x’. As another example, it is virtually impossible to include an entire o-ring motif in an ‘f’, ‘z’, or any of many other letter categories. The Adjudicator does not know how to excuse a letter from having to display a certain stylistic property, although the Letter Spirit program does gradually diminish the extent to which the Drafter tries to incorporate style in a letter when successive attempts at that letter have failed to generate a version of it that is recognized as such by the Examiner.

Drafter processing

The Drafter’s biggest limitation is that it cannot look ahead to detect the future implications of the individual steps that it takes. This could be solved by having it generate outputs tentatively, and offering those to the Examiner and Adjudicator for review before they are chosen by the Drafter. Ironically, rather than fundamentally changing the model, this would entail the same kind of processing already in Letter Spirit, but with the loop of review and revision taking place inside the Drafter. This would not be an amendment of the program’s functionality if the same powers of review in the current version of the program were merely considered to be in the Drafter rather than in the top-level loop and other modules.

A different consideration is that lookahead might not involve the grid, precisely. People can reach conclusions regarding general directions in drawing that are not quantum-specific, but depend on general geometrical notions of direction and grid boundaries. Planning that is not grid-specific is the whole idea behind the hypothetical Imaginer module (mentioned below), but a deeper understanding of two-dimensional space without any regard for the grid would yield great improvement if it were incorporated into *any* of the three existing modules.

Another perspective on lookahead is that experience while drawing letters probably leads to the automation of certain cautions and hesitancies that can come to bear in future drawing, so that problems can be flagged by certain familiar warning signs rather than elaborate consideration of the future consequences of decisions in drawing. This is doubtlessly part of letter rendering as performed by experienced humans.

The Drafter cannot explicitly aim to incorporate a motif into a gridletter in such a way that the motif would span two or more role-fillers, because it renders just one role-filler at a time.

People, unlike the Drafter, evaluate their decisions before committing to them, but that difference reflects how the Drafter is just one part of the larger Letter Spirit architecture and leaves evaluation for the other modules. It is difficult to say how well the Drafter's output compares to that of people because of this near-total lack of evaluation. The set of output coming from the Drafter thus cannot be said to be much like human attempts at gridletter creation.

The quantum-by-quantum nature of Coderack drafting produces an implicit "greedy" strategy wherein individual quanta that satisfy stylistic properties (often literal motifs) that are definitional of a given style are usually included in the first role-filler that is drafted. This can cause problems in cases of some particular combinations of letter, role, and style wherein drawing a given role first, imbuing it with the given style, makes it impossible to finish the gridletter in a satisfactory manner. In these cases, the program succeeds in creating acceptable output only when it selects the roles for drafting in precisely the right order, which may take place only one-sixth of the time, or even less frequently.

Letter Spirit

In the current revision strategy, each module creates or reviews potential output in units of one entire gridletter. One possible extension of the program that would be of great interest would be to give it the ability to review Drafter attempts and make the decision to approve or reject that output on the basis of units smaller than a gridletter (e.g., role-filler, or perhaps quantum). If some portion of a gridletter were found unacceptable, then it could be sent to a Drafterlike module that would *repair* the offending portion, while keeping the rest of the gridletter. The simulation of creativity in Chapter 4 showed that the smaller the grain size, the more efficiently that revision will work, so this could be a tremendous boon to the quality of output. This change might be made along with making the modules generally intertwine, so that any one module could run for a little while, as the program sees fit. This would call for a sophisticated model of top-level control, rather than the current simple loop.

The three-tiered regime of rejection, acceptance to the gridfont but not the style, and acceptance with promotion to seedhood is quite rigid. The ability to demote letters, and more generally, the ability to reclassify them freely and flexibly throughout a run, would certainly be more humanlike.

Currently, the Thematic Focus and the Library hold all the style information regarding the gridfont, and once a gridletter is added to the style, it is never reviewed again. A more realistic model of human design would allow the program to make style judgments about a gridletter that it has just designed by comparing it carefully on a one-to-one basis with relevant gridletters that have previously been accepted to the gridfont. For example, the review of the top of an ‘s’ might lead a future version of the program to turn its gaze specifically to the top of its version of ‘a’, if there is one, in order to compare them explicitly at that time.

A full-fledged Imaginer module could be added. If an Imaginer existed, it could benefit from a Conceptual Network imbued with feature nodes for high-level design decisions, like how much to weigh letter versus spirit, or how often to design post-and-bowl letters with role-sets that employ the circle role instead of the appropriate bowl.

The ability to build up the contents of long-term memory by learning is a desirable goal for any model of cognition. How the contents of the Conceptual Memory, the codelet types, and other aspects of the architecture might be learned are entirely open questions.

The sequential operation of the modules is not very realistic. As was noted earlier, people can begin evaluation of a gridletter before it is completely rendered (and before it is rendered on paper at all!). Additionally, people can flit around various portions of the task, as they find convenient. A person designing a ‘p’ may decide to re-examine the ‘b’ *during* the rendering of that ‘p’. The top-level program of Letter Spirit is quite rigid and does not allow meta-level decisions of this kind.

The Examiner can get the right answer for the wrong reasons (namely, with an anomalous parsing that by luck happens to lead to the categorization as a standard parsing). Such false recognitions look good in terms of raw percent-correct performance, but, because the rest of the program depends upon the parsing as well as the categorization, such events can lead to anomalous notions of style. This may be seen as a defect in the program, although it may be the case that this sort of quirk also happens in human gridfont design from time to time.

Three kinds of modes for promoting Drafter-designed gridletters to seed status were tested. However, each of those followed rather simplistic rules for making that decision; how people allow their work to change the direction of further work is surely quite complex and Letter Spirit does not model that in serious detail.

There are several inherent differences between letter categories that Letter Spirit, as it stands now, does not always recognize. A motif that is an o-ring, for example, is inherently easier to incorporate into some letter categories (like 'o'!) than into others. As another example, an abstract rule that suppresses diagonals places a modest constraint on a letter category such as 'l', but a very severe constraint on 'v' and 'x'. Such differences are not taken into account when style scores are calculated. For comparisons *within* letter category (determining which version of a given letter category is the best rendition of it), that shortcoming is not important, but for comparisons *across* letter categories (which letter category to select next for drafting, or what threshold to use as a cutoff in decisions regarding promotion to seedhood), this effect is insidious. No suggestions are offered here that would address this problem.

The program has limitations in its meta-level awareness. Ideally, it could realize new circumstances in a given run and alter its strategies accordingly. For instance, it might notice that in its current goal style, whole-gridletter borrowing by means of vertical flipping is unprofitable, and then avoid trying that strategy. Perhaps a future version of Letter Spirit could even have nodes representing strategies, which it could activate and deactivate as seemed appropriate during a run, allowing those activations to modify its strategy while a run is in progress.

Finally, although nondeterminism is a prerequisite for many of Letter Spirit's best characteristics, it introduces a wildness into the program that is hard to corral. The program is, in many cases, able to create good versions of gridletters for particular combinations of letter and spirit and yet fails to do so because its behavior is stochastic; it is never guaranteed to create any particular gridletter in any particular run. That Letter Spirit has the freedom to produced varied work means that it also has the freedom to produce flawed work. Its output is often scattershot. Letter Spirit will, for example, render the dot over an 'i' in such a way as to seem to show sensitivity to certain issues, but then render the dot over the 'j' in the same gridfont in a way that is not only different but markedly worse. To some extent, this weakness of haphazardness comes hand in hand with the strength of variability, but it seems that people are much better at attaining variability without producing so many anomalous creations. Letter Spirit is not only inconsistent in the level of quality of its output; it is also myopic with regard to things that it has done in the past and should do more of (or less of). Future work on Letter Spirit could strive to give the program more meta-level awareness and thus make it more humanlike in this way.

This section attests to the fact that there is no shortage of possible directions for future work on Letter Spirit. The full set of extensions recommended here is almost certainly beyond the scope of just one more dissertation. As exciting as some of the suggestions here are, the most exciting thing to come out of the next significant body of work on Letter Spirit might be the questions it leaves unanswered and the future directions that come into focus as long-range goals only at that point.

9.2 The FARG architecture

Letter Spirit stands as a work unto itself, but it borrows heavily from the general ideas behind the FARG architecture, first in a program named Jumbo and especially as it was realized in the Copycat and Tabletop programs, both of which are models of analogy-making (analogy, after all, being the 'A' in 'FARG') [Hofstadter 1983; Mitchell 1993; French 1992; Hofstadter and FARG 1995; Marshall 1999]. In many ways, Letter Spirit resembles those models; in other ways, it expands upon them; in still other ways, it simply diverges from them, owing in part to the differences between domains. This section seeks to place the work on Letter Spirit in the context of FARG programs, past, present, and future.

9.2.1 Is the Letter Spirit domain analogy?

Given the importance of analogy to the development of the approach underlying Letter Spirit, it is worthwhile to consider whether or not its task is analogy. Probably the best answer is that it is *like* analogy. One could consider the task of completing a gridfont, given, say, five seed gridletters, as an analogy in which one must answer "What is to the remaining twenty-one letter categories as these five gridletters are to their letter categories?" That question has the proper *form* to be an analogy problem, although compounding so many elements together on each side makes it unusual. Moreover, it does not state Letter Spirit's task precisely, because the sense of style can, through promotion, change throughout the course of a run. This is particularly true when Letter Spirit is given a smaller number of seeds, and it is most clearly seen in the case when *no* seeds are given to the program. Letter Spirit's task, first and foremost, is to produce a coherent gridfont, and giving it seeds is just one way to conduct a run (although it is the way most discussed in this thesis), and doing so only sets the initial goal style, in any case.

What the gridfont domain has most in common with analogy is that its task poses the program with an aesthetic problem — one for which there is never just one correct answer, although it is obvious that not all answers are plausible and not all plausible answers should be considered equally good. The program must be able to find a characteristic in its input (whether it be one side of an analogy or the style of some seed gridletters) and must be able to incorporate a characteristic (not necessarily the same one) into its output. By looking at human performance in both domains, one can see that the characteristics involved are frequently ones that are subjectively perceived; they are not amenable to rigid definitions and they are not easily incorporated into output by the application of any simple formula. Because of this, it is essential for programs working in these domains to represent concepts in a fluid manner. Fluid concepts have norms (such as the height of a post) that are preferred, but that can be slipped when the need arises. That concepts should be modeled in a fluid (the 'F' in 'FARG') way in both domains is perhaps the most important thing that the Letter Spirit task has in common with analogy.

9.2.2 Implementing the architecture

French [1992] lists a number of characteristics that are largely common to Copycat and Tabletop, and to their FARG successors, notably Jumbo and Seek-Whence [Hofstadter and FARG 1995]. The list was compiled by French after the implementation of those two programs, but resembles a list in [Hofstadter 1984]; it was, in effect, a wish list of prospective traits for future models of cognition. Letter Spirit can be seen as four different programs — the three modules and the top-level loop, with the potential for any and all to stand as representatives of the FARG approach. This subsection evaluates the extent to which each portion of Letter Spirit is faithful to its predecessors, using French's list (with one addition) as a basis for comparison.

Slipnet

Most FARG models make use of a memory structure called the Slipnet. Like conventional localist networks, the Slipnet consists of a set of nodes that represent concepts, and the nodes are linked together so that activation may spread between concepts that are related to one another. The added sophistication of the Slipnet lies in the way that each link has a length (the reciprocal of which is used to weight how much activation spreads through the link) that can vary throughout a run, reflecting how much the kind of relationship that the link describes seems to be important to the analogy in progress. For example, when "opposite" seems important, a node representing that concept will tend to be highly active, which causes the length of links such as that between "left" and "right" to become small, which in turn enables "left" to send a larger amount of activation to “right”, and vice versa. The name of the Slipnet is meant to convey the way that it facilitates jumps, or “slippages”, between one concept (such as “right”) in the source and a counterpart concept (in this example, “left”) in the target.

No part of Letter Spirit makes use of a Slipnet, although the Examiner has a standard localist network, the Conceptual Network. If Letter Spirit’s task is considered to be analogy, then the incorporation of stylistic properties into letterforms gives rise to the need for slippages; in the current implementation of Letter Spirit, these slippages are implemented not by a Slipnet but by a combination of several other devices, most notably by norm violations, which are incorporated into the target letterform during Coderack drafting. Slippages that amount to the inclusion of motifs and abstract rules are incorporated into the output subtly, particularly during Coderack drafting, but also in Library borrowing. To complicate matters, slippages in the output are not the responsibility of the Drafter alone, but are contingent upon approval by the Examiner and Adjudicator.

In some ways, then, Letter Spirit offers a picture of how slippage might occur that is more detailed than that seen in the earlier FARG models, because Letter Spirit has mechanisms that show the interplay between norms and proposed slippages on a lower level. This is particularly true of the Drafter and is true of the other two modules and even the top-level loop to a lesser extent.

In other ways, however, Letter Spirit is less sophisticated than earlier FARG models. Style is handled by its representation in the Thematic Focus (and the style of one gridletter at a time is handled by the stylistic properties in the Adjudicator’s Workspace), but there is no spreading-activation network, much less a Slipnet, that helps represent the importance, varying throughout a Letter Spirit run, of particular stylistic qualities. Decisions that are made by the top-level loop are handled in a relatively crude fashion, sometimes completely randomly, whereas a more humanlike handling would use high (and low) activations to promote (and suppress) certain techniques for drafting throughout a Letter Spirit run. To model gridfont design superbly, future versions of Letter Spirit should probably utilize Slipnets to help handle style and for top-level decision-making alike.

Coderack

Earlier FARG models had — as probably their most definitive trait — a Coderack rather than any central executive as the means of carrying out their activity. In a nutshell, each Letter Spirit module does the same, while the top-level loop does not. (Note also that the Drafter can render by borrowing, which does not involve a Coderack.) At present, Letter Spirit is more like a society of three FARGlike models acting in concert, with a central executive that coordinates them but that is not itself particularly FARGlike. Endowing (or replacing) the top-level loop with a Coderack-based central controller might be an interesting direction for followup work.

Long-term memory and short-term memory

Earlier FARG models had two kinds of declarative memory — a long-term memory (the Conceptual Memory) and a short-term memory (the Workspace). In a sense, Letter Spirit augments that two-tiered hierarchy of memory with a third tier. The Conceptual Memory, storing platonic descriptions of letters in terms of roles and ways of implementing style in terms of stylistic properties, is Letter Spirit's long-term store, and is shared by all the modules. Each module has its own version of the Workspace, which is thereby the program's short-term memory.

The *intermediate* level of memory that is novel to Letter Spirit consists of three memory structures. The style of the gridfont-in-progress is stored in the Thematic Focus and the Library, while the gridfont itself is stored in the Scratchpad (which, in terms of modeling human behavior, is probably best thought of as a virtual piece of paper rather than a mental representation). These are “read-write” kinds of memory (unlike the Conceptual Memory, which is immutable) and they can endure unchanged over time scales of more than a few seconds (unlike the Workspace). Letter Spirit inherently models a “larger” task than that of Copycat or Tabletop, and the extra level of memory reflects this greater complexity.

Continual interaction

Any of the earlier FARG models can be characterized in large part by the continual interaction between its Workspace and Conceptual Memory, as carried out by the model’s Coderack. Dynamically speaking, such a model shows relatively slow change on the higher levels of abstraction (the level at which output is generated), while frantic activity, flitting about from one concern to another with the execution of each codelet, takes place on a lower level; it is from the very small and independent steps of work done on the lower level that structure on the higher level gradually emerges.

This is largely true of the Letter Spirit modules as well, except that the intermediate level of memory is involved in Adjudicator and Drafter runs. During Examiner runs, there is indeed continual influence, on almost a codelet-by-codelet basis, of the Conceptual Memory, the Coderack, and the Workspace itself upon the Workspace, and by all of the above entities upon the Coderack. In Adjudicator runs, the Workspace is continually altered by activity involving the Coderack, the Conceptual Memory, the Library, the Thematic Focus, and the Workspace itself, but the Adjudicator also recommends prospective alterations of the Thematic Focus, which are sometimes carried out by the top-level loop. In contrast to other FARG models, codelets are posted onto the Coderack only in ways that are largely invariant from run to run. What has just been said of the Adjudicator is also true of the Drafter. Thus, these two modules differ from earlier FARG models in that they do not typically build up and tear down a substantial structure throughout a Coderack run in search of the complex structure that embodies their final answer; those modules do their work not in a single run, but through many runs of each module within a longer Letter Spirit run, and they do their building-up and tearing-down over the time scale of many runs rather than just one.

The top-level loop of Letter Spirit, if one chooses to view the modules it coordinates as black boxes carrying out fixed tasks, operates in a way much more coarse-grained than that of its constituent modules. In several ways, gross changes to memory structures take place abruptly: a new gridletter is added to the Scratchpad; dozens of small alterations to the Thematic Focus can be made rapid-fire; gridletters are rendered, rated, and live or die — all of these take place in a single "step". Seen this way, the behavior of the top-level loop seems far too coarse to be FARGlike. Of course, some of its "steps" are runs of modules whose internal behavior is perfectly fine-grained. What makes its overall behavior coarse-grained is that many of its steps are not fine-grained, and many kinds of interaction that could take place continually in human gridfont design cannot in Letter Spirit. As has been noted many times already, this could be corrected by interleaving the functions of the top-level loop's modules. Carrying out such a change to the program, however, is not straightforward, and would require significant future work.

Statistical, not deterministic, decision-making

FARG models make many of their decisions in nondeterministic fashion. In many cases, one decision emerges from the activity of many individual codelets. In other cases, a decision is made nondeterministically by the toss of a virtual coin; sometimes such decisions are maximally random (with all possible outcomes of equal probability); other times, such decisions are weighted. Even non-Coderack modes of the Drafter and the top-level loop are highly nondeterministic in their behavior.

Nondeterminism has countless benefits, not all of which will be reiterated here.

The most obvious one is that a strategy of review and revision is of no use whatsoever if all attempts at the same output are identical. Nondeterminism means that a program yields different output in different runs, and the variety in the output demonstrates the character of the program far better than the single possible output that would come from a nondeterministic program. Variety in behavior is thus not only humanlike, and not only necessary for Letter Spirit's central strategy to work, but is also a source of the rich corpus of output whose analysis makes it possible to understand the program and how it models human cognition.

Parallel terraced scan and temperature

All FARG models make use of the parallel terraced scan, and in most instances temperature is used to modulate the extent to which more outlandish possibilities are considered. For Copycat, Tabletop, and Letter Spirit's Examiner, this happens with the numerical calculation of temperature being used to influence the extent to which codelets' urgencies affect the choice of codelets from the Coderack. The other components of Letter Spirit also employ the parallel terraced scan, but in ways that are different; those ways have already been described in earlier chapters and summarized in this one.

9.2.3 Parallels with a contemporary

Almost all of the work on Letter Spirit took place after Copycat and Tabletop had been implemented and described in published form. Another FARG model, however, was in development in the same time frame as Letter Spirit. Metacat, the dissertation work of Jim Marshall, is a follow-up to Copycat with the aim of solving the same kinds of analogy problems that Copycat works on, but with more sophisticated means of working on them [Marshall 1999].

Like Letter Spirit, Metacat is concerned with imbuing a program with the ability to monitor its own work and to modify its efforts in accord with what it sees. Metacat, however, was built by incorporating new features into Copycat, maintaining the structure of one Coderack-driven program rather than, as was the case with Letter Spirit, by building a larger framework within which several Coderack-based modules are run.

Because each program monitors its own work, each performs actions on (at least) two time scales. On a quicker time scale, predominantly Coderack-driven action flits and whirs about, leading to actions such as the creation of a gridletter or the choice of an answer to an analogy problem. On a slower time scale, each program evaluates such actions and may, in one way or another, either approve of such work, disapprove of and retract such work, or approve of such work and resolve to use characteristics of that work in future work. Each program has memory structures that serve the self-evaluation process, and, as each program seeks to capitalize on and perpetuate *themes* in its own behavior, each program has a memory structure whose name reflects this — the Thematic Focus in Letter Spirit and the Themespace in Metacat. (Letter Spirit also uses the Library and the Scratchpad to support the general goal of reviewing its own work.) Because self-watching is only found, among FARG models, in these two programs, those memory structures are used in ways quite different from any memory structures in other FARG models. A typical FARG model includes some memory structures (for example, the Workspace of most any FARG model) that are accessed and altered by the Coderack very frequently and others (for example, the Examiner's Conceptual Memory) that are inalterable long-term stores. The two self-watching programs have memory structures that are altered, but only occasionally, as self-watching behavior cuts in when enough work has been done to merit an examination of what has been done since the last episode of self-watching. The significance of this "extra" (relative to earlier FARG models) level of memory is that there is an extra level of behavior being modeled, a level of self-awareness on which themes in behavior, rather than simply behavior itself, is the focus. The earlier FARG models, having done a reasonably good job of modeling certain aspects of behavior, paved the way for Letter Spirit and Metacat to attempt, by means of similar strategies, to model a higher level of human behavior.

Despite the important similarities between them, the two programs do differ in interesting ways. Metacat's approach to self-watching differs from Letter Spirit's in that Letter Spirit simply looks at its output and takes action based upon that, while Metacat preserves a record of its past activity (this is one of its forms of intermediate-level memory) and subsequently tries to improve upon unsatisfactory output by taking into account what processes led to it.

Metacat and Letter Spirit represent two distinct approaches to extending the FARG repertoire of modeling techniques. Both approaches involve some kind of self-watching involving memory structure which is altered by the program, but only on the cognitive level of activity. That similarity is probably owed to the power of self-watching and the inherent need to have such a memory structure in order to make it work. The differences in their approaches show that there is more than one way to skin a cat.

9.2.4 The greatest FARG strength

FARG models, individually and cumulatively, have many positive characteristics of many different kinds. This section will close by identifying one strength that is shared by all FARG models and that has a particularly satisfying explanation in terms of a phenomenon mentioned earlier in this thesis.

In Chapter 4, it was shown that deriving an output of a given size (such as a gridfont) and a given level of quality could be expedited by selecting a smaller grain size (such as a gridletter) and working to improve the product on that level, combining the parts into a better whole. This is much of the essence of what FARG models do. The analogy-solving FARG models do it by working on the parts of an analogy problem and producing in each run their final answer based upon structures built in a bottom-up fashion in the perception of the input analogy problem. When the problem has been perceived deeply, resulting in structures that represent the problem as the program has seen it, then the proposed answer follows in straightforward fashion. The Examiner does it by working first on the level of labels, then that of roles, then that of role-sets. The Adjudicator does it by working first with stylistic properties, and then deriving style from them. The Drafter does it by working first with types of sugar (pressures used to influence drafting) and then choosing a quantum based upon them. Letter Spirit as a whole does it by starting with individual gridletters and ending up with a gridfont.

Task decomposition is hardly original to FARG models. Their unique strength, however, is in undertaking creative tasks while directing most of their processing power on a lower level (a subcognitive one), deriving the benefits of decomposition (achieving good output faster) but still doing just enough work on the higher level to guide processing and to ensure that the quality of the output is good. It is easy, when considering FARG models, to forget the well-worn idea of AI as being search and nothing more. An observer thinking of all AI as search, and considering how large the possible search spaces of answers to the problems that FARG models tackle are, might wonder how a FARG model can spend so little time searching such a large space and nonetheless manage to find a good answer. In essence, it is the work done on the lower levels that makes the answer come to the search instead of requiring the search to go find it.

9.3 Conclusion

One thing that is obvious from all that has been said about Letter Spirit up to this point is that there is a lot to say about it! It is a very big program, and its runs are very long and complex. A typical Letter Spirit run totals about 1.4 million codelets, easily hundreds of times longer than runs of other FARG models. At this point, the commentary that has been generated regarding the program has also grown very long.

This section will offer two final ways of addressing Letter Spirit as it now stands by looking at the program in terms of questions and goals that were posed before it had been implemented. In addition, Chapter 10 will give a final retrospective on the project by discussing the task of writing the program, rather than the program itself.

9.3.1 Skepticism from an unlikely source

The author of the present version of Letter Spirit first read of the project in the 1980s [Hofstadter 1985], but heard more about it in talks by Gary McGraw on the Indiana University campus in the autumn of 1992. At that point, I felt that the task of implementing Letter Spirit was hopeless, that the essences of letter and of spirit were far too slippery to be grasped anytime soon. I envisioned someone having a miserable experience trying to make such a thing work, and certainly did not imagine that I myself would have any involvement in the project. Later, when the Examiner was being developed, and I had accepted the vague responsibility of eventually taking the project over, I made a list of concerns regarding the project. With the implementation of a working version of the program complete, these questions can now be answered.

How will the program have a memory of mistakes so that it can avoid repeating mistakes made within a given run? Answer: Within the Examiner, it does not. It is hard to say that the Adjudicator makes identifiable mistakes at any specific point in its task. The Drafter is prone to repeating mistakes made in Coderack drafting but will only attempt a certain method of whole-gridletter borrowing once per run. Finally, the top-level loop remembers, during each run, each attempt at a gridletter that has been created and how it was rated; however, this does not prevent the Drafter from wasting its efforts by re-creating gridletters (good or bad) that it has already created once before.

Will the program avoid combinatorial search? This seems particularly hard to do in searching for motifs. There are no combinatorial explosions in Letter Spirit runs, slowing them down; motif-searching is indeed the trickiest thing to handle without spending a lot of time in search. In principle, such searches could involve combinatorially-many steps. The code in Letter Spirit avoids such problems in a few ways, mainly helped out by the fact that motifs tend not to be very large.

Will the idea of levels of enforcement (how important a stylistic property is to a style) be represented with a single number or will it emerge in some other way? This ended up being the levels of the Thematic Focus. Essentially, it is a number attached to each stylistic property, one that changes when promotion or demotion occurs. Future work might make the representation of this idea more lively by allowing the level of enforcement of a stylistic property to change even in the middle of what is now an Adjudicator or Drafter run. A person designing or evaluating a gridletter can pause in the middle and review other gridletters and weigh the relative importances of various aspects of the style. For now, Letter Spirit lacks this ability.

How are the various modules and subtasks to be coordinated? Do they run in arbitrary order? The structure of Letter Spirit runs is fairly firm and explicitly built into the top-level loop. The fact that subtasks of Letter Spirit cannot be ordered in more flexible ways is the more general shortcoming behind the one mentioned in the answer to the previous question.

Are concepts that the program notices appropriately bound to other concepts? As one example, if a poor 'k' causes high temperature, is that temperature bound to the *'k' so that other letters do not get "blamed" for it? As a second example, if a stylistic property such as slanting is noticed in posts alone, can slanting somehow be bound to posts so that bowls and other roles do not necessarily end up with slanting in them?* Poor ratings for a gridletter are attached to that gridletter in the Scratchpad; no other gridletters end up getting blamed for another's faults. Stylistic properties are not attached directly to specific roles, but specific role-fillers (and all imaginable properties of them) are stored in the Library, so that borrowing may transfer those properties over to other letters using the same roles. A relative norm violation often ends up being *de facto* specific to a particular role, or a small set of roles, because the norm violation is defined in terms of a property (such as "very-tall") that is not common to most other roles. This, together with Library borrowing, has much the effect of binding particular stylistic properties to particular roles. An enhanced ability with which to imbue the program might be to give it the notion of *sets* of roles, such as "posts" and "bowls", so that it may, in appropriate circumstances, treat the entire class in a similar manner.

Overall, none of the problems that I saw in 1992 and 1993 were too serious. The thorny issues raised in my five questions were all dealt with, one way or another, in my implementation, even if better solutions await future work. Most importantly, the project did not prove, as I expected in 1992, to be an unsolvable problem fated to mire a graduate student in futile attempts to achieve the impossible.

9.3.2 The end of the beginning

A great deal was written about Letter Spirit when its implementation had not yet begun and when it was still in its earliest stages. Already, a core of excellent ideas had come forth from the *project*, waiting to become part of the *program*. The hopes for the program that were expressed at that stage concerned whether or not it would be capable of *making its own decisions* and thereby accomplish the goal of creating gridfonts that were stylistically coherent. With a particular emphasis on the goal of modeling decision-making and having creativity emerge from it, it was written:

> It is our fervent hope to realize a program of this degree of complexity and subtlety. It remains to see how far we can actually carry it. [Hofstadter and FARG 1995]

Perhaps I am biased for having seen the program run, but is my firm belief that the program shows that degree of complexity and subtlety. It still remains, as it always will, to see how far we can carry it.

CHAPTER TEN

The Meta-Art of Cognitive Modeling

10.1 Introduction

In the most general terms, the goal of this thesis is to inform the reader about how the work done on Letter Spirit contributes to the field of cognitive modeling and other related fields of inquiry. An understanding of the program's mechanisms and behavior — the topics of the nine previous chapters — is obviously central to that goal. However, cognitive modeling is not only the study of the mind, but also the practice of a craft — that of constructing cognitive models. Like all crafts, practicing it requires not only an understanding of the objects produced, but also the procedures involved in the production. The goal of this chapter is to discuss not Letter Spirit itself but the work that went into its implementation. For some readers, who will do work on other models of cognition, this information may possibly be as useful as the particulars of the other nine chapters. As cognitive models may create art of their own, cognitive modelers create artists, and so this chapter is about the meta-art of cognitive modeling.

It is not possible in one mere chapter, or even one volume, to provide a comprehensive treatise on the methods of developing cognitive models; that is a burgeoning field in its own right. The aim of this chapter is simply to describe specific matters that came up in the process of programming Letter Spirit.

10.2 Means and ends

It is not unheard of in cognitive science for one researcher's work to strike another researcher as being of questionable value. This is surely due, in part, to the fact that different researchers have different goals, and one person's whole career may entirely neglect the goals that another person finds essential. This section tries to lay out how work on Letter Spirit relates — or fails to relate — to some other kinds of work. While the difference of focus from one researcher to the next is often on which aspects of cognition are studied and modeled, the emphasis in this section is on broad methodological issues.

10.2.1 Was this science?

This work was officially conducted for a degree in the fields of computer science and cognitive science, but the work that was done was not always particularly scientific. Science's usual concerns with controlled experimentation fell by the wayside — and necessarily so. The program is simply too big for each of the features in it to have been tested thoroughly for cognitive plausibility. If the weights attached to norms in the Conceptual Memory are counted as parameters, then the total number of parameters in Letter Spirit is a few thousand. (It is credible that some lines of code in Letter Spirit have actually never been involved even once in any run of Letter Spirit that has yet been carried out.) This fact alone shows that the program could hardly have been implemented at all — and certainly not by just two or three people — if it had been considered essential to experimentally verify that each feature in the program matched up with characteristics of human behavior. An even better indicator that an arduously scientific approach would not have been practical comes from the interdependence of the modules. This is particularly true of the Adjudicator and the Drafter. It is impossible to test either of those modules thoroughly without having the other one on hand to use in the prospective tests. This introduces a chicken-and-egg problem in which neither can be verified as fully "correct" until the other one is. Given those observations, it should not be surprising that hardly any of the parameters in Letter Spirit were set according to the results of careful experimentation and that all of the modules have characteristics that came from guesswork and intuition, but have not been verified as characteristics of human gridfont design.

Indeed, not only were many of the fine details of the program not determined by careful application of the scientific method; some major features of the model were not, either. The general structure of the Thematic Focus, for example, was not based on a careful fit to human cognition. The central goal was to allow for some items in a representation to carry more weight than others, and for the weight assigned to each to be able to vary over time. Up to that level of detail, the Thematic Focus matches the characteristics of human behavior it is intended to model. Other details, such as the idea of discrete levels and promotion and demotion based solely on frequency of occurrence, are well intended, but arbitrary. The best that can be said of those design decisions is that it is hoped that they do not deviate too far from human behavior, but they are surely not highly accurate renditions of their respective facets of human cognition.

In brief, some aspects of the program were based upon careful experimentation, but many others were not. Most of the broadest and most central aspects of the program are corroborated by ideas from cognitive science that were, in turn, based upon scientific experimentation, although at least one broad aspect of the program —

that the modules run one at a time — is not grounded in empirical findings, and is, in fact, almost certainly not humanlike. Details of the program on lower and lower levels are generally less likely to be based upon empirical work or the literature and more likely to be based upon introspection or simply a hunch that the chosen mechanisms would make the program work.

Science is, to say the least, highly thought of in academia, and particularly in fields that use the word in their name. Consequently, the fact that a model of cognition should be based (in part) on anything besides science is bound to be a source of concern to some. These qualms surface in Skinner's opinion of cognitive science as a whole:

> I accuse cognitive scientists of relaxing standards of definition and logical thinking and releasing a flood of speculation characteristic of metaphysics, literature, and daily intercourse, speculation perhaps suitable enough in such arenas but inimical to science. [Skinner, 1987]

The rest of this section will argue that work on Letter Spirit is work well worth doing, not only in spite of, but also in some ways because of, having adopted something other than a purely scientific stance.

10.2.2 Was this engineering?

In a grudge with the toughest kid on the block, it could be useful to have another tough kid on one's side. So, it could be claimed that work on Letter Spirit was an engineering effort, a single-minded drive to create a program that worked, and that it was for that reason that the work on Letter Spirit did not employ science as its sole guide.

If that were the reason why work on Letter Spirit has not been guided solely by empirical findings, it would be a sufficient one. Many impressive technological feats have been undertaken without the kind of step-by-step rigor, peer review, etc., that accompany scientific research. In fact, much like Letter Spirit, many engineering projects would require an inordinate amount of time (perhaps boundless) to subject each detail of them to empirical justification.

To say that Letter Spirit is engineering rather than science is not entirely wrong, but not entirely right, either. Engineering strives to solve problems without particular concern for the means chosen. The Letter Spirit project has not proceeded under the belief that gridfont design is an abstract task that has meaning apart from the human beings who undertake it from time to time. Rather, modeling human behavior is precisely the goal, and learning about human behavior is the intended result.

It is true that anything short of a working program would have been a highly disappointing result. Without a working program, there would have been no output or behavior to evaluate and to compare to human work on the same task, and no way of evaluating at all which parts of the model were cognitively plausible and which were not. For that reason, some aspects of the program reflect the need, when there was no clear indication of how to make the program more humanlike, to make sure, first and foremost, that the program did carry out the task somehow or other. In this respect, it often was engineering that replaced science as a motivation, but usually only in regard to the lower-level details.

10.2.3 What was this?

In essence, the work on Letter Spirit was an effort to combine well-established work in cognitive science with software engineering in order to create a product (the program) that could test hypotheses about cognition. The ideal course of action, in which every minuscule detail of such a program were verified for authenticity through meticulous empirical work, is simply impossible to undertake. The result is a program that embodies a certain mixture of theory, speculation, and details that are arbitrary with regard to theorizing about cognition but are nonetheless functional.

It is presumably appropriate to concede that the ideal course would have made it easier to evaluate the behavior of the model, and when it reached the level of being ready to run, a verified and comprehensive theory of creative behavior — and therefore of much of cognition as a whole — would be complete. With that sort of approach being currently — if not permanently — unavailable, the kind of methodology that was used in developing Letter Spirit is preferable to not doing the work at all or waiting forever for the details to be put into place meticulously so that an effort to put those details together into a computational model could happen sometime in the next century or later.

Much of this thesis has identified — with inconstant degrees of certainty from one topic to the next — which aspects of the program likely make a good match with the corresponding mechanisms in humans and which aspects deviate from the goal and need to be addressed in future work. It is hard to imagine that a painstaking devotion to Skinner's standards could do so much to advance the understanding of the processes underlying creative behavior in so short a time. Without a doubt, the methodology underlying this work and a number of other efforts in cognitive science has found merit in relaxing the standards that have long ruled science precisely because approaches that are so inflexible have consistently proved less fruitful in studies of cognition than in the disciplines upon which science cut its teeth.

10.3 Design decisions

The description of Letter Spirit's mechanisms hints at the work that was done to create them, but leaves unspecified many of the details. It is not possible to describe the work done on Letter Spirit in full detail, but a discussion of some of the thorny issues that plague the work and some of the considerations that motivated final design decisions may be useful to those who design and adapt computational models of cognition.

10.3.1 Development: Examiner-heavy

When implementation began in 1993, there was no formal plan that stipulated exactly how much time would be spent in the development of each of the different parts of Letter Spirit. Nor were there precise specifications of what each part of the program would accomplish, although there was a broad overview [Hofstadter and FARG 1995]. But overall, the constraints defined in advance were loose, and so the amount of time spent on each module and the precise nature of each module were free to evolve while work on the program was being done. Eventually, because of a few important priorities, the Examiner ended up almost monopolizing the effort put into development, with each successive part of the program receiving less attention than the one before it. The tale of how development proceeded and how it was loosely scheduled with imperfect foresight is perhaps instructive.

It was decided in the early 1990s that Gary McGraw would undertake implementation of the Examiner module for his dissertation work. The detailed programming work began in earnest in 1992. By early 1994, the code had proceeded to the point that some gridletters (initially, just 'b's and 'd's) could be recognized. By early 1995, the Examiner could perform its intended task fairly well, and it was tweaked only slightly before Gary's dissertation work came to an end later in 1995.

During 1995, just before Gary ended his programming work on Letter Spirit, I began my work on the program, but I focused initially on the graphics aspects of the code (which, along with other interface aspects of the program, required a fair amount of work, although that work has hardly been mentioned at all in either this or Gary's thesis). In the first half of 1996, in discussions with Doug Hofstadter, several options were considered for how work should proceed at that point, including whether to delve into the Adjudicator, to quickly complete a version of the program that designed new gridletters by means of borrowing and what came to be called the Library, or to return to the Examiner and enhance it in certain ways. Eventually, the work on the Examiner came to the forefront. This was simply motivated by a desire to increase the speed of the Examiner; all other aspects of the performance of the Examiner were perceived as satisfactory. However, it was noted that if each Examiner run lasted several minutes, then a run of the whole Letter Spirit program might take days. In addition, the length of Examiner runs made it difficult to proceed with development of other portions of the program because they all depended upon the Examiner, and successful development requires many runs of the program to test and debug the code. Slow run times thus lead to even slower development. Making the Examiner faster thus became the central goal, and work on that optimization (as it was called in Chapter 5) went on until about the end of 1996.

In early 1997, I turned my efforts to the Adjudicator, and the notion of sprinting to a quick closing-of-the-loop using a borrowing-only means of creating new gridletters was put aside. Several months of programming created a prototype of the Adjudicator that could identify the stylistic properties in a gridletter once it had been parsed by the Examiner. In the summer of 1997, however, Doug and I found the output of the Examiner–Adjudicator combination to be unsatisfactory because it was not capable of detecting many of the most important norm violations in gridletters given to it. As was discussed in Chapter 5, this was diagnosed as being due to the set of which roles and role-sets were available to the program at that point, and it was clear that only an overhaul of the Examiner could fix the problem. That work consumed several months, with the outcome being an Examiner that was faster, better in terms of percent of correct answers, and — most important — satisfactory in creating parsings that allowed the Adjudicator to produce the desired range of norm violations — by January of 1998.

Subsequent work focused on the Adjudicator, the Drafter, and the top-level loop, respectively. Even in that time span, however, small alterations still needed to be made to the Examiner occasionally, as the routines I had implemented for segmenting gridletters crashed the program in rare circumstances that showed themselves only when extensive testing using strange gridletters (so strange that people never created them, but the often-quirky Drafter did) was underway. The last Examiner change was not complete until February of 2000, only shortly before work on the program as a whole was completed in April of that year. Although the order in which work on each of the four major parts of the program began was Examiner, Adjudicator, Drafter, top-level loop, the order in which work on each of them ended was Adjudicator, Drafter, Examiner, top-level loop.

In brief, the story of this section is how the Examiner came to dominate development on Letter Spirit (and not because it was done poorly initially). The Examiner is, simply put, the bedrock upon which every other part of the program must be built. It is the only one of the four parts that could, in principle, be implemented satisfactorily without even a rudimentary version of any of the other three parts being in place. In practice, as this section has shown, getting the Examiner just right depended upon having near-final versions of every other part of the program. The general lesson is perhaps that hindsight can be so superior to foresight that it is worth sprinting to a tentative version of an entire program and finding flaws in the parts as development proceeds rather than trying to craft satisfactory versions of each part of a program one at a time.

10.3.2 Roles and role-sets

It was clear early on that the Letter Spirit project called for letters to be defined, graphically, in terms of a collection of roles and role-sets that would be hard-wired into the program's Conceptual Memory. While we felt strongly that roles were the correct way of representing letters, we did not have any strong theory for why any one way of defining each letter in terms of particular roles was the correct one. For example, was an 'f' composed of two roles (a hooked post and a crossbar) or three (a straight portion of the post, the crossbar, and a separate role for the hook)? Should both letter-conceptualizations be put into the code, or only one of them? Or was there a third way of dividing 'f' (a fourth, a fifth?!) that we were overlooking? Around the beginning of 1994, Gary McGraw and I agreed one day that we would both separately sketch a way of doing this satisfactorily for each letter and that by comparing the two versions, we would see whether or not there were any controversial choices to be made after more involved thought. Our renditions of the alphabet seemed similar enough[1] that we decided that it was not needed to give the matter much thought beyond picking one of the sketches or some combination of them to use in the implementation.

As this matter has already been discussed at length in Chapter 5 and was mentioned again in this chapter, it would belabor the point to say more than that our choice of roles and role-sets led to subsequent problems. It is perhaps accurate to say that a correct choice in 1994 would have required no more effort on anyone's part at that time and would have saved six months of developing effort in 1998. Perhaps a smirk from a Skinnerian would be deserved at this juncture. A more cautious, scientific approach would have proved the adequacy of a collection of roles and role-sets before proceeding and thus, in theory, this sort of problem would have been avoided. It is certainly true, though, that that level of caution and care, applied to every other aspect of the program, would have made implementation time-consuming to the point of intractability. In a sense, every point in design is a gamble, in which caution bears the sure cost of time and effort, but may save more time and effort later. While role and role-sets were an instance in which the gamble did not pay off, it clearly did in

[1]Of course, we had been talking about letters for months beforehand, so our two-pronged attack naturally did not yield two totally independent approaches.

many other instances, as well as in the aggregate.

10.3.3 The numbers problem

The inner workings of Letter Spirit, it has already been noted, contain countless hard-wired values for parameters and numerous functions that compute values that vary from run to run based upon other values that they take as input. Those functions themselves contain numerous coefficients, adding to a staggering number of tiny design decisions that had to be made, with not very much time devoted to each such decision. This suggests a consequential risk that some of those decisions would be made badly, leading to poor performance and a poor fit to human behavior.

Experience soon taught me that many of the places in the code where such decisions had to be made were extremely forgiving of haphazard choices. For example, the possible widths of a left-post, as defined in the code now are 'skinny' (a width of zero) with a weight of 10, 'half-wide' (one quantum wide) with a weight of 9, and 'wide' (two quanta wide — as wide as the whole grid) with a weight of 6. How sensitive is the program's performance to those precise values? By and large, not very. If the weight for 'wide' were made 5 or 7, or if the weight for 'half-wide' were made 8, the only effects would be that some parts would, during the Examiner's process of sparking, have their numerical match with the 'left-post' role changed by 1 point out of a potential total of 210. It would be quite rare that that single point in the role definition would make the difference in whether or not a given part sparked 'left-post'. Consequently, no great care was taken in determining precise values of this kind.

It *is* quite important, though, that weights fall within reasonable bounds. Changing the weight for 'half-wide' in the previous example to 11 *would* have important (and undesired) ramifications, because that would make 'half-wide' weighted the highest of the three possible widths and thus the width norm for left-posts. The Adjudicator would then find skinny left-posts as norm violators and the Drafter would tend to design the majority of its left-post role-fillers as wider than normal, and Letter Spirit would be slightly quirky, relative to human gridfont designers in making its left-posts wider, on average, than the humans'.

The instances of design decisions in the code are too numerous to describe in full, but the example of left-post width communicates the primary lesson here, which is that each design decision provides the designer with a certain (and varying) amount of slack. The first thing to know is how much slack a situation affords. Then make a choice within those bounds. Especially when the bounds are large, the decision may be arbitrary within them with no appreciable impact on the fidelity of the program as a model of human behavior.

10.3.4 Statistics as an aid to design

While many of the nitty-gritty details in Letter Spirit were chosen by the intersection of theory and educated but somewhat arbitrary fiat, a couple of major classes of details were built into the code by something akin to machine learning. The prototypes used by the Examiner's gestalt codelet were produced by statistical summaries of large numbers of gridletters. A set of such summaries had already been used for purposes of analysis in 1993, and was added to the Examiner's code during the 1996 optimization. In addition, some of the weights attached to norms in the Conceptual Memory were calculated in a similar way. Because of the vast number of parameters that needed to be entered in this way, it was viewed — during the 1998 Examiner overhaul — as excessively laborious to complete the entire process by hand. A large set of gridletters was pre-segmented by hand, then a routine that gathered statistics kept track of, for example, how many left-posts were "tall", how many were "medium-height", and so on. These frequencies were turned into weights such that the modal value for each role along each feature dimension (such as height, weight, etc.) received a weight of 10, with lower frequencies scaling to lower weights. These lists were then edited by hand to make proper adjustments when necessary (for instance, some portions of the code require that there be just one modal value for each role in each dimension, so that only one height of left-post is weighted 10; second place must be 9 or less).

Besides serving as a time-saving device for the developer, this method of weighting norms for roles was a hedge against forgetfulness on the part of the programmer because a large enough sample of input gridletters encompasses variety that helps define reasonable expectations for how tall, wide, etc., roles actually can be without forgetting unusual but allowable variations that come up rarely. While explicitly tailoring role definitions to handle a number of quirky example gridletters is helpful for producing a system with robust performance, it introduces a possible problem, which is discussed next.

10.3.5 Testing with the training set

Making sure that a program works well over a given input set is not intrinsically a bad thing. On the surface, it seems purely a good thing. However, if the test set used to validate the behavior of the program consists entirely or in large part of a training set, and if painstaking care was involved in assuring that the program performed well on that training set, then the program's performance on the test set is less impressive. If it is known in advance what the desired behavior of a program is for each item in a potential test set, then in principle a large table could simply store the desired answers. A table of that sort would not shed any light on the processes underlying cognition.

While Letter Spirit clearly does not consist of a mere table of responses to a list of possible inputs, the concern that one might have is that a less blatant version of the same thing might exist — that Letter Spirit's behavior in a number of situations is appropriate because the program was specifically tailored to handle those situations. The primary concern is not that the *training* set was inappropriately chosen, because the program did not come about so much from machine learning as from cycles of programming and testing. When changes were made to the program, the new version of the program would be tested, and the quality of the behavior would determine whether or not the changes were good ones. Thus, it is not a training set but a *test* set used at the time of development that is the subject of concern, because those tests, for the sake of expediency, were almost invariably far smaller in scope than the full set of tests reported in this thesis for the final version of the program. The problem would arise, then, if the development-time test set were essentially the same as the test set used in the final analysis of the program, and would thus provide no independent confirmation that the processes in the program were general ones.

By analogy, consider someone learning English who only knew a few words. If that vocabulary consisted only of basic words such as "hello", "please", and "me", then a listener would quickly get a fair assessment of the speaker's ability. If that vocabulary contained a disproportionate number of low-frequency words such as "lamprey", "oscillate", and "millstone" (like the hypothetical program trained on the test set, this vocabulary would surely be the product of a feat of memorization), then an utterance made by that person would suggest — contrary to fact — that they must have a rather large vocabulary. In the case of a gridfont-designing program, the issue at hand is not how many gridfonts can be handled, but that those that can be handled typify a class of gridfonts that are comparable to or simpler than the test set in terms of complexity. One wishes to infer that the behavior that is demonstrated by runs on test sets is *indicative* of the program's ability, and not a series of flukes that are rare, isolated examples of the few things that the program can do. So, if the program could handle only Standard Square, House, Boat, and other styles of similar simplicity, a broad set of tests, by exposing the program's inability to handle other gridfonts, would inform the observer that the program was limited in that way. However, if the program could handle Standard Square, Three-D, and Sluice, but nothing else, a test showing those successes but not showing any failures could lead to the mistaken conclusion that the program had great range in its abilities.

Most of the development-time tests on my programming work on Letter Spirit involved various subsets of the gridfonts Standard Square, Shorts, House, Benzene Right, Benzene Left, and Hint Four. The objective was to make sure, at each important stage along the way, that the program could handle basic motifs, abstract rules, and norm violations. The test set of 23 gridfonts used in this thesis extends sufficiently beyond the development test set in terms of sheer size and in terms of variety of styles to make the abilities and limitations of the program in trials beyond its development-time test set clear.

10.3.6 Norm dimensions: Two's company or a crowd?

In the optimization and overhaul of the Examiner, some of the dimensions used to represent norms of roles were brought into question and, in some cases, changed. A good example is that of the representation of roles' vertical span. In the 1995 Examiner, this was done with a set of elementary features such as 'baseline–midline', 'baseline–top', 'baseline–bottom', 'baseline–t-height', and 'baseline–x-height'. Thus, these were values on a dimension that might be called "vertical span", with each feature indicating the top *and* bottom (not always in that order in the names, although the orthography of the names had no effect on the program's behavior) of the role. (As always, the role was defined with weighted lists of such values, with the norm in each dimension being the highest-weighted value.)

Subsequent work on the Examiner, aimed at facilitating the Adjudicator's handling of norm violations, made it highly desirable to revamp this way of representing where the top and bottom of a role or role-filler should ideally lie. Consider that there are seven possible places where the top *or* bottom of a part may lie. That means that there are dozens of possible combinations of where the top may lie and where the bottom may lie (some of those possibilities less likely than others, to be sure). A norm violation might involve any possible combination of them. For example, a short ‘t’ could contain a norm violation that replaces the norm ‘baseline–t-height’ with ‘baseline–x-height’. A straightforward implementation of the code that could calculate such norm violations would be tedious, with literally hundreds of possible norm violations, and hundreds of conditions to check to see which one applied in a given situation.

An elegant way to solve this problem is to represent the locations of a part’s highest point and its lowest point as values in two different dimensions (both measuring vertical position, but with one measuring the vertical position of the role-filler’s top and the other measuring the vertical position of its bottom). With that scheme, there are only seven possible values in each dimension (the same seven, of course) and the calculation of a norm violation is a simple comparison of one value with another in a linearly-scaled dimension — unlike that originally used — in which it is easy to perform arithmetic (in terms of what is higher and what is lower). Of course, it would be possible to maintain the original representation and convert to the bidimensional approach, perform the comparison, and convert back, but why? In this case, representing the characteristic of interest is best done with two dimensions, not one that systematically crams two pieces of information into one label.

In contrast, a similar, but not quite identical, case arose in the case of the tips of roles and role-fillers. Again, in the original 1995 Examiner, one dimension coded what might be seen as a pair of features — namely, the location and the direction in which it pointed. Typical values were ‘down-center-baseline-tip’, ‘left-slash-baseline-tip’, and ‘right-baseline-tip’. In this case, it might again seem desirable to calculate norm violations for location and for direction separately. Again, the issue arises of a maddening number of possible permutations from one value to another if both kinds of property are coded in a single dimension. And once again, utilizing two dimensions instead of one solves the problem.

However, in the case of tips, the combination provides a crucial piece of information. This is because location and direction of tips, while they may be able, in principle, to covary freely, tend to have a higher degree of relation to one another. The case of 'v' provides an excellent example. The left-wing of a 'v' can reasonably have its upper tip at any of the three points at the x-height, or, if the left-wing hooks around counterclockwise, at the point in the middle of the left side of the central zone. That tip can also reasonably point in any direction from east to north to west, or any value in between, or — if the left-wing hooks around as mentioned before — down to south. But, significantly, the permutations of location and orientation do not covary freely. A southwest orientation is apt if the role-filler hooks around and then down (as is the case with the 'v' in the gridfont Snout), but is very unlikely if the tip is on the x-height, and even less likely if the tip is on the x-height and in the middle of the grid horizontally (as with the 'v' of Weird Arrow). Somehow, the program needs to be able to recognize that the eccentricity of slipping both of those norms (location and orientation) is not merely the sum of how eccentric it is to slip one plus how eccentric it is to slip the other. If the dimensions were kept separate, that is precisely how the mathematics would proceed. Thus, it was essential to have the added control that comes with binding what could, in principle, be two dimensions into one.

By analogy, barnyard animals may moo or cluck, and they may give milk or eggs, but those variables do not covary freely. The simpler representation has the locus of the range of variation tied up in one dimension (species), with the characteristics of noisemaking and edible product dependent upon an animal's value in the species dimension. In another instance, like representing a person's height and eye color, it makes sense to have two dimensions, rather than to combine the two and have values like 'tall-and-blue'. The general design decision here is to note whether or not it is preferable to bind two dimensions together.

10.3.7 Dependence, independence, and development

The Drafter is the counterpoint to the Examiner, as the part of Letter Spirit that has thus far received the least attention in its development. It was functioning fairly well by April of 1999, after only weeks of initial work on it. The time it took to get the module in decent working order was astonishingly short compared to that required for the Examiner and the Adjudicator. However, the Drafter was still somewhat erratic in its output, and a phase of tweaking went on late into September of that year. In contrast, the Adjudicator had a long period of initial development, but comparatively little tweaking took place once the program had first begun to work. The two modules differed vastly in terms of the ratio of how much time was required to get them to work at all and how much time was required to tweak them to work more or less as desired.

There are several reasons why the development of the two modules would have such different profiles in terms of this ratio (after all, their tasks and structure are fundamentally different!), but the one of interest here is that certain characteristics of the Drafter are more highly interrelated than any equivalent set of characteristics of the Adjudicator. In particular, the Drafter's performance depends critically upon the weights that determine how much each type of sugar influences quantum-by-quantum drafting.

Each type of sugar involved in Coderack drafting amounts to a score that is calculated for each candidate quantum that may be the next to be drawn. These calculations are themselves potentially quite complex. In the last, but long, period of tinkering in the Drafter's development, however, the calculations were all more or less fixed except for the set of coefficients, or weights, used to determine how much each one would be able to influence the nondeterministic selection of the next quantum (if any) added to the role-filler being drafted. As the programmer, I had the power to make one sugar type's weight higher and thereby increase its influence or to make it lower and diminish its influence. The goal was to find a set of coefficients that led to the desired behavior — namely, somewhat erratic in the way that Drafter output was meant to be, but converging upon humanlike prioritization of the various kinds of role norms and stylistic properties.

This ended up being a rather excruciating and time-consuming process, because any change that solved a problem in one particular situation seemed to create a new problem for some other situation. This led to a regular mode of activity in which a problem would be noticed, a solution would be attempted, the problem would be solved, and a subsequent test for some other combination of role, letter, and gridfont would indicate another problem; then the "solution" would be undone, and the cycle would begin anew.

Unfortunately, this problem had no good solution. It amounted to search of a poorly-understood multidimensional space. Moreover, the evaluation function (how the program with a given set of coefficients performed over a sizable development test set) was very slow to compute by the standards of how fast would be ideal. Instead of a test of a new set of coefficients taking place in seconds or less, it could take a day or more. I watched the test runs unfold, looking for inspiration that could guide me to a satisfactory set of coefficients, but progress was nonetheless slow, and thus took months to conclude. The Drafter's performance was not terrible at the beginning of that tinkering phase, but the increased performance was probably worth the development time required to achieve it. This decision, like many others, involved a tradeoff between a desire for quality and a desire to keep the time spent in development reasonable.

10.3.8 Codelets as types vs. tokens

In the Examiner optimization, I sought any means possible to speed the Examiner. In the terminology of economics, I saw each computation the program performed as an expenditure, and my desire was for the stingiest budget possible. One place that I detected waste was that certain codelets were placed on the Coderack with parameters representing parts bound to them. For example, a top-glommer codelet would be put onto the Coderack with a specific part in the current segmentation as the part that it might go on to combine with one of its neighbors. A frequent occurrence in Examiner runs was that some codelets of that kind would be on the Coderack at the time that a re-segmentation occurred. When such a codelet happened to be executed later, its operand, the part bound to it when it was posted, would no longer exist, and thus the codelet would "fizzle" (in other words, be removed from the Coderack but otherwise do nothing itself). Although this was not a large waste of time (a codelet that does nothing does not take very long to run), this quickly caught my eye as an expenditure that could be done away with.

A modification I had in mind would be to have no codelets ever take bound parameters. Instead, whenever a codelet was selected that required an operand, an operand would be selected from among those that existed *at the time the codelet was selected for execution.* This would eliminate any codelets from being "wasted". In addition, the selection of the operand could be done in a principled way, in keeping with FARG fashion, done by a nondeterministic choice weighted by whatever considerations seemed relevant.

Although this would not have been grossly at odds with the FARG approach, it was argued against by Doug Hofstadter and Gary McGraw, and was ultimately not implemented. The reason can be explained in terms of the discussion of representations that bind two dimensions into one. The original (and implemented) way of binding parameters into codelets was a sort of recommendation that the codelet in question eventually run *acting on the bound parameter in question.* The proposed amendment to the program would break down that philosophy by having codelet types and their parameters covary freely, which could perhaps go counter to the program's reason for posting the parameter in the first place. That is illustrated by the earlier example of the top-glommer codelet. The posting of a top-glommer codelet expresses that there is a perceived problem with the current segmentation. If a re-segmentation takes place, then that codelet *should* go unexecuted rather than being executed with some other poor part as its victim. In general, this is an argument for codelets on the Coderack to be thought of as codelet *tokens* rather than codelet *types*. Thinking of codelets on the Coderack as tokens rather than as types has been the FARG approach so far and was proven the better approach in this instance.

10.3.9 Motifs

The final example offered of a design decision will be the choice of the scheme used for the representation of motifs in Letter Spirit. It was quickly obvious (see the discussion of example gridfonts' styles in Chapter 1) that it would not be sufficient to see two shapes as examples of the same motif only if they occupied, in different gridletters, the exact same location on the grid. However, it was equally clear that if a shape *did* always occupy one specific location on the grid, then that would be an important thing to remember about that motif. The same consideration also applied to motifs that were invariant as well as those that varied, respectively, in terms of orientation and reflection, and all possible combinations of translation, rotation, and reflection.

In the early planning for this, I found an elegant set of representations, one for each of several types of motif. Literal motifs (allowing no change in location or orientation) would be stored as lists of quanta such as "5, 36, 20". Translatable motifs would be stored as sequences of compass-point headings such as "west, southwest, south". Rotatable motifs would be stored as sequences of angles (in degrees), such as "+135, +180, –45". Finally, rotatable and reflectable motifs would be stored as sequences of angles with their sign specified in a relative way so that each set of them could easily flip signs in the rendering and matching processes, such as "+/–135, +/–180, –/+ 90". Each of these could be stored concisely and compared quickly with potential matches in an obvious way.

In discussions between Doug and me, the concern was raised that having only four classes of motif did not do justice to the potential places along the continuum of literality. The four types I had initially described all existed on that continuum, to be sure, but so did many other possibilities, such as motifs that could be rotated by 180°, but not by all increments of 45° or those that could be reflected horizontally but not reflected vertically or rotated at all. It became clear that people were capable of distinguishing perhaps a dozen or more levels of literality. This called for enriching the types of motifs that Letter Spirit could distinguish. In the end, five types (described in Chapter 6) were utilized rather than four, distinguishing the importance of not only literality and translatability, but also rotation in increments of 45°, 90°, and 180°. Though falling well shy of the dozen or so (or more) ways that people can treat motif literality, this did enrich the class of motif types enough to improve performance significantly.

It should be noted that no elegant way of representing this new set of motif types was found (despite considerable pursuit of such a representation). As a result, matching of rotatable motif types came to involve numerous comparisons as the motif to be matched is rotated through the requisite number or positions — as many as 16. The tradeoff that was accepted slows the Adjudicator and the Drafter down significantly, perhaps by a third or a half, but it constitutes a gain in cognitive plausibility. That should serve as the final reminder in this thesis that the primary objective of Letter Spirit was to capture human abilities, not simply to solve an engineering problem.

10.4 An architecture as a meta-tool

The focus of this thesis and this work has primarily been on the level of the Letter Spirit project, one instance (or a few, depending upon how one looks at it) of the FARG architecture. The design-decision process detailed thus far in this chapter has similarly focused upon the specifics of the Letter Spirit program. The central tenets of the architecture have already been discussed. This final section considers the magnitude of the task of starting a FARG model in a new domain from scratch.

10.4.1 Phases vs. Coderacks: Serial and parallel operations

Phases became a part of Letter Spirit beginning with the 1995 Examiner's regime of roles' being loosened in discrete steps, one for every several hundred codelets run. The idea of phases was adapted to other parts of Letter Spirit, particularly the Adjudicator. Building phases into the structure of a program is complementary to the usual FARG Coderack-only (or Coderack-mainly) approach because it enforces order and seriality as opposed to nondeterminism and implicit parallelism, which have already been noted to be essential traits of Coderack processing.

The Adjudicator is the Letter Spirit module most structured by built-in phases, with three phases that are each of relatively unvarying length from one run to the next, and that always occur in a fixed order. The Examiner has a greater number of phases, but phases there only involve the manipulation of one global variable (the extent to which roles are loosened), which only subtly moderates the behavior of the program rather than leading it by the nose on a new course, and even the value of that one global variable influences behavior only when a sparker codelet nondeterministically decides not to use the original level of loosening. Drafter Coderack runs have no phases at all (or, they could be seen as one phase per run); that simplicity is related to the fact that an individual run of the Drafter Coderack only does a small portion (usually the choice of one quantum) of the bigger task of drafting a gridletter. A whole run of the Drafter, then, usually containing several Coderack runs within it is like an Adjudicator run in having several distinct phases of Coderack processing organized by a higher level of structure that is not Coderack-driven. Yet another approach is taken by the top-level loop of the program, which has no Coderack processing at all and is thus essentially all phases.

If phases are defined to be the mode of processing in which the sequence of execution is predetermined by the programmer, then the overwhelming majority of all computer programming, whether in cognitive modeling or otherwise, has adopted that methodology. It is the contrasting use of Coderack-style processing that is a big part of the original contribution that FARG models have made. However, work on Letter Spirit found it useful, in the ways described already, to mix in some rigidly sequential phases to accomplish certain things. Perhaps future work will find ways to switch modes of operation with a design based purely on Coderacks rather than built-in phases.

10.4.2 Codelet design: Serial and parallel operations

One of the first things that anyone carrying out any sort of algorithm (think of a recipe) learns is whether or not the order of the steps matters. On the surface, it would seem that Coderack architectures, with their nondeterministic selection of the codelet to be run next, are banking on the fact that the order does not matter at all. However, this is not true. The Examiner, in extremely low-probability circumstances, could fail to work properly at all in a certain run if the selection of codelets were repeatedly unfortunate, perhaps because one or another type of codelet was consistently overlooked so that no codelets of that type ever got to run. This is extremely unlikely, somewhat like a person suffocating because all the oxygen molecules in the room randomly but systematically traveled to other places within the room. Instead, rather than being an impediment to the Examiner, the low-level disorganization leads to variety in its recognition; thus, two attempts to parse a gridletter, even with identical segmentations, can lead to different attempts at an answer. This variability can lead to the Examiner finding a good answer when the correct answer is not the first possibility tried, or in any sense the most obvious possibility to explore.

In some cases, though, there is not necessarily a compelling reason to "Coderackize" a set of steps. The first phase of the Adjudicator finds stylistic properties in a gridletter. The sequence in which they are found does not matter in any way whatsoever. In terms of the final outcome of the run, one might just as well scrap the Coderack approach for that phase and have a serial procedure look for stylistic properties in a fixed order. This is perhaps true for other phases of the Adjudicator, for the entire Drafter, and for portions of the Examiner as well. In the optimization of the Examiner, a codelet for checking r-roles, one role-set at a time, was eliminated on the grounds that having that operation was found to be not only not helpful, but also even actively harmful (for reasons described in Chapter 5).

One may find the Coderackized nature of those portions of the program superfluous. I think members of FARG may argue that it is nice to have the nondeterminism and implicit parallelism going on in underlying processing, to reflect that we believe that it is a realistic model of how people behave, even if it has no possible effect on the final answer. While the discussion in this thesis has largely ignored the possibility of runtime displays, they have been an important part of FARG models (particularly those that have fewer than one million codelets per run). It is not only the final result of a run that matters; the way that a FARG model's attention flits about from one possibility to another has been intended to be part of what is being modeled as a human characteristic.

10.4.3 An architecture as a meta-meta-tool

A great deal of the progress that has been made in the world of computer software has come from the gradual building-up of levels upon other levels, providing the users and developers with powerful tools without their needing to reinvent more basic levels: assemblers, compilers, libraries, and developer's toolkits based on graphic user interfaces make programming tasks that might otherwise be impractical relatively easy. The question has periodically come up whether or not creating FARG modeling could be helped along by a software toolkit that provides a developer with some ready-made framework into which the "mere" details of a new model need to be inserted, thus making the creation of FARG models easy. In the extreme, one might then view the FARG paradigm as a programming language. At least three salient examples exist — the Prolog programming language, SOAR [Newell 1990], and ACT-R [Anderson and Lebiere 1998] — in which a piece of software that began as a theory of thought turned into a sort of franchise of cognitive modeling, if not a full-blown programming language.

It is trivially the case that *some* effort could be made to create a framework that would facilitate the programming of future FARG models. The question is how much of the burden can be lifted from the shoulders of future modelers. My experience with Letter Spirit itself suggests that the amount that effort can be reduced is fairly small. The amount of domain knowledge itself is an enormous part of Letter Spirit. It contains a theory of letters that was distinct from any previous FARG model. This dictated the contents of the Conceptual Memory (which are rather vast in terms of sheer volume) and the memory structures that needed to be created (especially consider the Thematic Focus). The only codelet type that *could* clearly be common to potential implementations of, say, Copycat and the Examiner is the activation-spreading codelet (and as it turns out, not even that one *was* shared, as Copycat handled activation spreading in a different way rather than with a codelet!); the other codelet types had to be conceived of and implemented in a manner specific not only to the domain but to the theory (of roles) and the approach (labeling and sparking). Even with much in common between the Letter Spirit modules, none of the three share a common codelet type. Clearly, the majority of the effort that must go into any new FARG modeling project of the level of complexity of those undertaken thus far is work that must be specific to the domain in question, and cannot be bypassed by making use of a software library consisting of general modeling tools.

The central software element that can clearly be passed on from model to model is the Coderack. The routines that post, select, and run codelets could be used in anyone's Coderack-based architecture. At least one portion of the formula in the Letter Spirit code for this had its origin in Tabletop code. This is reminiscent of the old British naval tradition of removing one plank from a ship as it was being decommissioned and placing it into a new ship for a symbolic continuity across the ages even as the fleet had to be continually maintained and modernized. Perhaps this or other portions of Letter Spirit's code may be of use in future FARG models, but at this point, this body of work on Letter Spirit has come to an end, and the future lies with posterity.

REFERENCES

Abes, C. (1994). Expert graphics: Graphics professionals share their secrets. *Macworld*, 11(8):122.

Adams, D. (1986). *A Dialogue of Forms: Letters and Digital Font Design.* Master's thesis, Massachusetts Institute of Technology, Cambridge, Mass.

Alley, T. R. and Cunningham, M. R. (1991). Averaged faces are attractive, but very attractive faces are not average. *Psychological Science*, 2(2):123–125.

Anderson, J. and Lebiere, C. (1998). *Atomic Components of Thought.* Erlbaum, Hillsdale, NJ.

Aristotle, trans. by Wardman, A. E., and Creed, J. L. (1963). *The Philosophy of Aristotle*, *Poetics*, pages 413–431. Penguin, New York.

Barratt, K. (1994). *Logic and design: In art, science and mathematics.* Lyons Press, New York.

Barrett, T. (1994). *Criticizing art: Understanding the contemporary.* Mayfield Publishing, Mountain View, California.

Biederman, I. D. (1987). Recognition by components: A theory of human image understanding. *Psychological Review*, 94(2):115-147.

Binsted, K. (1994). Computer generation of linguistically definable riddles. Technical Report Technical Paper 25, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland, 1994.

Birkhoff, G. D. (1933). *Aesthetic Measure.* Harvard University Press, Cambridge, Mass.

Blesser, B., Shillman, R., Cox, C., Kuklinski, T., Ventura, J., and Eden, M. (1973). Character recognition based on phenomenological attributes. *Visible Language*, 7(3).

Boden, M. A. (1990). *The Creative Mind: Myths and Mechanisms.* Basic Books, New York.

Boselie, F. and Leeuwenburg, E. (1984). A general notion of beauty used to quantify the aesthetic attractivity of geometric forms. In Crozier, W. R. and Chapman, A. J., editors, *Cognitive processes in the perception of art.* Advances in Psychology (19). North-Holland, Amsterdam.

Bouma, H. (1971). Visual recognition of isolated lower-case letters. *Vision Research*, 11:459–474.

Cave, K. R. and Wolfe, J. M. (1990). Modeling the role of parallel processing in visual search. *Cognitive Psychology*, 22:225–271.

Chamberlain, W. (1984). *The Policeman's Beard is Half-Constructed: Computer Prose and Poetry.* Warner Software/Books, New York.

Christianson, K., Hollingworth, A., Halliwell, J., and Fereira, F. (2000). Thematic roles assigned along the garden path linger. Submitted for journal publication.

Coe, M. D. (1992). *Breaking the Maya Code.* Thames and Hudson, New York.

Cope, D. (1991). *Computers and Musical Style.* Oxford University Press, Oxford.

Dennett, D. (1991). *Consciousness Explained.* Little, Brown, and Co., Boston.

Desimone, R. and Duncan, J. (1995). Neural mechanisms of selective visual attention. *Annual Review of Neuroscience*, 18:193–222.

Desimone, R., Miller, E. K., Chelazzi, L., and Lueschow, A. (1995). Multiple memory systems in the visual cortex. In Gazzaniga, M., editor, *The Cognitive Neurosciences.* MIT Press, Cambridge, Mass.

Ebert, R. (2000). *Roger Ebert's Movie Yearbook 2000.* Andrews McMeel Publishing, Chicago.

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.

Falkenhainer, B., Forbus, K. D., and Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63.

Falkenhainer, B. (1990). A unified approach to explanation and theory formation. In Shavlik and Dietterich, editors, *Readings in Machine Learning.* Morgan Kaufmann, San Mateo, California.

Fechner, G. T. 1925 work trans. by Niemann, M., Quehl, J., and Höge, H. (1997). Various attempts to establish a basic form of beauty: Experimental, aesthetics, Golden Section, and square. *Empirical Studies of the Arts*, 15(2):115–130.

Forbus, K. D., Gentner, D., and Law. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science* 19(2):141–205.

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

Fricke, H. (1981). *Norm und Abweichung.* Beck, Munich.

Geyer, L. (1977). Recognition and confusion of the lowercase alphabet. *Perception and Psychophysics*, 22(5):487–490.

Gibson, E. (1971). Perceptual learning and the theory of word perception. *Cognitive Psychology*, 2:351–368.

Gibson, E. J. (1969). *Principles of Perceptual Learning and Development.* Meredith Corporation, New York.

Gilmore, G., Hersh, H., Caramazza, A., and Griffin, J. (1979). Multidimensional letter similarity derived from recognition errors. *Perception & Psychophysics*, 25(5):425–431.

Goldstone, R., Medin, D., and Gentner, D. (1991). Relations, attributes, and the non-independence of features in similarity judgments. *Cognitive Psychology*, 23(2):222–264.

Goldstone, R. and Medin, D. (1994). The time course of comparison. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20:29–50.

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

Grush, R. (1995). *Emulation and Cognition.* PhD thesis, Univeristy of California, San Diego.

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)* II, 127–132, Washington, D.C.

Hennessy, D. and Hinkle D. (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert*, 7(v):21–6.

Hinton, G. and Sejnowski, T. (1986). Learning and relearning in Boltzmann machines. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing, Volume 1.* MIT Press, Cambridge, Mass.

Hockett, Charles F. (1960). *The View from Language*, pages 124–43. University of Georgia Press, Athens.

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

Hofstadter, D. (1983). The architecture of Jumbo. In *Proceedings of the International Machine Learning Workshop*, Monticello, IL.

Hofstadter, D. (1984). The Copycat project: An experiment in nondeterminism and creative analogies. AI Memo 755, MIT Artificial Intelligence Laboratory, Cambridge, Mass.

Hofstadter, D. (1985). *Metamagical Themas.* Basic Books, New York.

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

Höge, H. (1995). Fechner's experimental aesthetics and the Golden Section hypothesis today. *Empirical Studies of the Arts*, 13(2):131–148.

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

Holland, J. H. (1986). Chapter 20: Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine Learning: An Artificial Intelligence Approach*, volume 2, pages 593–623. Morgan Kaufmann Publishers.

Pincus, J. W., Turner, B., and Johnson, A. F. (1986). *The Encyclopaedia of Type Faces.* Blandford Press, New York.

Johnson-Laird, P. (1988). Chapter 8: Freedom and constraint in creativity. In Sternberg, R., editor, The Nature of Creativity, pages 202–219. Cambridge University Press, New York.

Jones, T. (1995). *Evolutionary algorithms, fitness landscapes and search.* PhD thesis, University of New Mexico, Albuquerque.

Kanigel, R. (1991). *The man who knew infinity: A life of the genius Ramanujan.* Washington Square Press, New York.

Kahneman, D. and Miller, D. (1986). Norm Theory: Comparing reality to its alternatives. *Psychological Review*, 93:136–153.

Kant, I. (1977). A theory of aesthetic judgement. In Dickie, G., and Sclafani, R. J., editors, *Aesthetics : A critical anthology*, pages 643–687. St. Martin's Press, New York.

Keren, G. and Baggen, S. (1981). Recognition models of alphanumeric characters. *Perception & Psychophysics*, 29(3):234–246.

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

Koch, W. A. (1963). On the principles of stylistics. *Lingua*, 12:411–22.

Koestler, A. (1975). *The Act of Creation.* Picador, London.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things.* University of Chicago Press, Chicago.

Langley, P., Simon, H., Bradshaw, G., and Zytkow, J. (1987). *Scientific Discovery: Computational Explorations of the Creative Process.* MIT Press, Cambridge, Mass.

Langlois, J. H. and Roggman, L. A. (1990). Attractive faces are only average. *Psychological Science*, 1:115–121.

Lauer, David A. (1979). *Design basics.* Holt, Rinehart and Winston, New York.

Leeuwenburg, E. (1971). A perceptual coding language for visual and auditory patterns. *American Journal of Psychology*, 83, 307–349.

Lenat, D. (1982). AM: Discovery in mathematics as heuristic search. In Davis, R. and Lenat, D., editors, *Knowledge-Based Systems in Artificial Intelligence*, pages 1–225. McGraw-Hill, New York.

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

Luce, R. D. (1963). Detection and recognition. In Luce, R. D., Bush, R. R., and Galanter, S. E., editors, Handbook of Mathematical Psychology, Volume 1. John Wiley and Sons, New York.

Lynch, D., edited by Rodley, Chris (1997). *Lynch on Lynch.* Faber and Faber, Boston.

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

McClelland, J. and Rumelhart, D., eds. (1986). *Parallel Distributed Processing.* MIT Press, Cambridge, Mass.

McClelland, J., McNaughton, B., and O'Reilly, R. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–57.

McCorduck, P. (1991). *Aaron's Code: Meta-art, Artificial Intellgence and the Work of Harold Cohen.* Freeman, New York.

McGraw, G., Rehling, J., and Goldstone, R. (1994). Letter Perception: Toward a conceptual approach. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society.*

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

Machotka, P. (1995). Aesthetics: If not from below, whence? *Empirical Studies of the Arts*, 13(2):105–118.

Maltin, R. (2000). *Leonard Maltin's Movie and Video Guide 2000.* Signet, New York.

Marr, D. (1982). *Vision.* Freeman, San Francisco.

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

Meehan, J. (1976). *The Metanovel: Writing Stories by Computer.* PhD thesis, Yale University, New Haven, Connecticut.

Meggs, Philip D. (1998). *A History of Graphic Design.* John Wiley and Sons, New York.

Minsky, M. (1985). *The Society of Mind.* Simon and Schuster, New York.

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.

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

Nanard, M., Nanard, J., Gandara, M., and Porte, N. (1989). A declarative approach for font design by incremental learning. In Andre, J. and Hersch, R., editors, *Raster Imaging and Digital Typography*, pages 71–82. Cambridge University Press, Cambridge.

Newell, A. (1990). *Unified Theories of Cognition.* Harvard University Press, Cambridge, Mass.

Nöth, W. (1995). *Handbook of Semiotics.* Indiana Univeristy Press, Bloomington, Indiana.

Nozaki, A. (1990). *Kasparov versus Deep Blue: Computer chess comes of age.* Springer, New York.

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

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

Pirsig, R. M. (1974). *Zen and the Art of Motorcycle Maintenance.* Bantam Books, New York.

Plato, trans. by Grube, G. M. A. (1974). *Republic.* Hackett Publishing Company, Inc., Indianapolis, Indiana.

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.

Poincaré, H. (1892-99). *Les méthodes nouvelles de la mécanique céleste.* Gauthier-Villars, Paris.

Polya, G. (1954). *How to solve it.* Princeton University Press, Princeton, NJ.

Posner, M. I. and Petersen, S. E. (1990). The attention system of the human brain. *Annual Review of Neuroscience*, 13:24-42.

Posner, M. I. (1995). Attention in cognitive neuroscience: An overview. In Gazzaniga, M., editor, *The Cognitive Neurosciences.* MIT Press, Cambridge, Mass.

Quart, A. (1999). Font punks. *Artbyte*, 2(2).

Rehling, J. (1996). Multiple representation types in perception In *Proceedings, Seventh Midwest Artificial Intelligence and Cognitive Science Conference.*

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

Riffaterre, M. (1959). Criteria for style analysis. *Word*, 15:154–74.

Riffaterre, M. (1971). *Essais de stylistique structurale.* Flammarion, Paris.

Ritchie, G. and Hanna, F. (1990). AM: A case study in AI methodology. In Partridge and Wilks, editors, *The Foundations of AI: A Sourcebook.* Cambridge University Press, New York.

Rowe, J. and Partridge, D. (1991). Creativity: A survey of AI approaches. Technical Report R 214, Department of Computer Science, University of Exeter, UK.

Russell, S., and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach.* Prentice-Hall, New York.

Samuels, A. L. (1967). Some studies in machine learning using the game of Checkers. *IBM Journal*, 11:601–617.

Sanocki, T. (1986). Visual knowledge underlying letter perception: Font-specific, schematic tuning. *Journal of Experimental Psychology*, 13(2):267–278.

Schacter, D. L. (1987). Implicit memory: History and current status. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13:501–518.

Schank, R. C. and Childers, P. (1988). *The creative attitude: Learning to ask and answer the right questions.* MacMillan, New York.

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, Mass. 1990.

Schmidhuber, J. (1998). Facial beauty and fractal geometry. Technical Report IDSIA-28-98, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Manno, Switzerland.

Searle, J. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3:417–457.

Shepard, R. N. (1978). Externalization of mental images and the act of creation. In Randhawa, B. S. and Coffman, W. E., editors, *Visual Learning, Thinking and Communication*, pages 165–211. Academic Press, New York.

Sheppard, A. (1987). *Aesthetics: An introduction to the philosophy of art.* Oxford Univeristy Press, New York.

Shirer, W. L. (1960). *The Rise and Fall of The Third Reich.* Simon and Schuster, New York.

Chase, W. G. and Simon, H. A. (1973). Perception in chess. *Cognitive Psychology*, 4:55–81.

Singh, D. (1993). Adaptive significance of female physical attractiveness: Role of waist-to-hip ratio. *Journal of Personality and Social Psychology*, 65:293–307.

Skinner, B. F. (1987). *Upon Further Reflection.* Prentice-Hall, Englewood Cliffs, NJ.

Sloman, S. A. (1996). The empirical case for two systems of reasoning. *Psychological Bulletin*, 119:3–22.

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

Strickland, C. and Boswell, J. (1992). *The Annotated Mona Lisa: A Crash Course in Art History from Prehistoric to Post-Modern.* Andrews McMeel Publishing, Chicago.

Todorov, T. (1965). *Théorie de la littérature. Textes de formalistes russes.* Seuil, Paris.

Townsend, J. (1971). Alphabetic confusion: A test of models for individuals. *Perception & Psychophysics*, 9(6):449–454.

Turvey, M. T. (1973). On peripheral and central processes in vision. *Psychological Review*, 80:1–52.

Treisman, A. and Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*, 12(1):97–136.

Ullman, S. (1989). Aligning pictorial descriptions. *Cognition*, 32:193–254.

Van Gelder, T. J. (1997). Dynamics and cognition. In Haugeland, J., editor, *Mind Design II.* MIT Press, Cambridge, Mass.

Vecera, S. P., and Farah, M. J. (1994). Does visual attention select objects or locations? *Journal of Experimental Psychology: General*, 123(2):146–60.

Vitz, P. C. and Glimcher, A. B. (1984). *Modern Art and Modern Science: The Parallel Analysis of Vision.* Praeger Publishers, New York.

Wittgenstein, L. (1953). *Philosophical Investigations.* Macmillan, New York.

Zemsz, A. (1967). Les optiques cohérentes (La peinture est-elle langage?). *Revue d'Esthétique*, 20:40–73.