PHAEACO:
A COGNITIVE ARCHITECTURE
INSPIRED BY BONGARD'S PROBLEMS

Harry E. Foundalis

Submitted to the faculty of the University 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

May 2006

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

_____

Dr. Douglas R. Hofstadter
(Principal Adviser)

_____

Dr. Michael Gasser

_____

Dr. Robert Goldstone

_____

Dr. David Leake

Bloomington, Indiana
May 2006.

For my brother, Dimos E. Foundalis,
who guided my thought since the beginning of my life

Στον αδερφό-μου, Δήμο Η. Φουνταλή,
που καθοδήγησε τη σκέψη-μου από τα πρώτα-μου βήματα

ACKNOWLEDGMENTS

Looking as far into the past as my memory allows, I discern a few milestones that led to the creation of this project. Each milestone is associated with someone who played a vital role in my ability to realize the present work.

The first event happened when I was a child in Greece, in sixth grade. According to my father, I was having trouble expressing my thoughts in a patterned, standardized, and rather stilted way for a course called “Exposition of Ideas” (akin to “Composition” in the U.S. education system). So he bought me a book that contained dozens of examples of essays, from which I was supposed to learn how to write. I do not believe I learned anything from that book; with only one exception, all its essays were utterly boring. But the one exception was about the brain and our mental world, and I found it absolutely captivating. I read it from beginning to end several times, and became extremely eager to learn more about the ways our “self” is created out of the ceaseless functioning of billions of neurons. It was the first time I stumbled upon the idea that all our “subjective experiences”, all that constitutes our rich, ethereal, and private mental world, might be the outcome of an entirely non-mental, materialistic layer, made of biological components and amenable to scientific scrutiny. It was also the first time I remember myself thinking: “When I grow up, I would like to have a job that will let me learn more about how the mind works.” The more succinct version of this thought, “I want to become a cognitive scientist”, couldn’t have occurred to me back then, since the term “cognitive science” had not yet been coined.

The second milestone is not exactly an "event", since it spreads over a period of 15 to 20 years. I associate it with my brother, who, being ten years older, was able to guide my interests to match his own. Besides inspiring my broad interests in scientific matters, and reinforcing my conviction that the world is utterly explicable in terms of what our senses allow us to perceive, my brother, quite unconsciously, instilled in me the ability to construct *algorithms*. With numerous construction-projects he taught me to be like him: careful, systematic, aware of the impact of every fine detail, persistent, and undaunted by failure. Later, in my college years, the notion of an algorithm became explicit for both of us. I ended up possessing a programmable Texas Instruments scientific calculator, and we both spent countless hours programming the little hand-held device in its arcane, but so well-designed machine language. This continued when, in 1987, we both acquired our first "real computer", an IBM PC/XT, capable of being programmed in the then-fashionable language Turbo Pascal. By that time I realized I had a certain knack for designing algorithms, but it was only later that I understood and appreciated the abstract and very fundamental role my brother played in molding my character.

After earning a Master's degree in computer science, in 1993, and working for four years for a software development company, I finally realized my interests were not in commercial enterprises, but in science. I remember trying to find, for hours on end, the solution to a problem: given a collection of planar points that form approximately a curve, how do we find the curve that approximates them? What if the points form not just one, but several curves? How does our eye solve this problem? Little did I know that this problem had been solved long time ago, but my desire to find an algorithm was intense. My wife played a pivotal role in letting me understand that I would benefit from continuing my studies for the

Ph.D. degree. So it was decided that I would apply for a doctoral program in computer science in the U.S.A. Two years later, in September 1995, I entered Indiana University with a double major in computer science and cognitive science.

Within two years I had figured out a way to represent conceptual relations. I drew some rather detailed figures of the abstract ways in which concepts should be connected, but had not implemented these ideas in any way. At the same time, coincidentally, I was reading Douglas Hofstadter's *Gödel, Escher, Bach*: *an Eternal Golden Braid*. Upon reaching page 652, to my utter astonishment, I saw my own figures printed! Well, they weren't exactly "my figures", but it was such a close match that I thought something mysterious was happening. But there was no mystery: my thoughts had simply converged roughly to the same representational scheme that Hofstadter had envisaged more than 20 years before. This made me very curious about Hofstadter's work. By a second fortunate coincidence, Hofstadter was physically in Bloomington, a member of the faculty in computer science, cognitive science, and various other departments at Indiana University. I took a class with him, and in less than a year I was a member of his research team, and he became my advisor.

Hofstadter's work, philosophy, and approach in cognitive science made a very deep impression on me, and marked the direction of my subsequent research. Phaeaco, described in this thesis, bears the unmistakable signature of a "Hofstadterian" architecture. In addition, I had the opportunity to use the facilities of his research lab (the Center for Research on Concepts and Cognition, or CRCC) for eight consecutive years, with few obligations beyond my own work. This was a very generous arrangement that few graduate students have the opportunity to enjoy. I am forever grateful to my advisor for this.

Albeit for different reasons, each of the three other members of my Thesis Committee deserve a warm “thank you”. Michael Gasser, for letting me interact and exchange ideas with the students of his own research group (GLM), attending their weekly meetings for over five years, and giving me numerous pieces of advice on several occasions; Robert Goldstone for kindly lending me the facilities of his laboratory in the psychology building, without which my experiment on human subjects would have never been completed; and David Leake, for giving me early guidance, including a term project on visual perception in one of his classes in computer science, which became the initial stepping stone on which Phaeaco was eventually based. An initial stub of a very small but essential routine in Phaeaco (called “Process B” in chapter 10), was written by my collaborator in that project, Etsuko Mizukami.

Some of my friends played an indispensable role in the development of this work. My colleague, Maricarmen Martínez, deserves a special mention. It was often after long discussions with her that many of my ideas about cognition crystallized, and various aspects of the project took shape. Maricarmen was my sounding board, and continues to offer essential moral support. Alexandre Linhares was also a source of great encouragement, and often of undeserved but always appreciated compliments. John Rehling, the designer of *Letter Spirit* (one of Hofstadter’s projects), provided both his spirit in several lively discussions years ago, and his “letter” (his dissertation) — a solid guide on which I based the present one. Yoshiko Osawa was my Japanese language helper for a linguistics project that began in the present work, but has not yet been completed. I should also mention that I used Yoshiko’s low-quality printer to scan the Bongard problems from Bongard’s book, and resisted all temptations ever since to re-scan them at a better resolution, reasoning that since our human eye can deal with them

at this poor quality, my program should be able to do so as well. Finally, Ralf Juengling, a friend and collaborator for one year, ported Phaeaco's C++ code to Linux, and gave me many useful suggestions for improvements. He was devoted to the project, and worked meticulously for months on end, completely undaunted by the complexity of the task. His advisor, Melanie Mitchell (the author of Copycat), offered me the opportunity to present Phaeaco at the Oregon Graduate Institute in Portland, Oregon, in an invited lecture.

Of the other people who contributed in various ways to this project I cannot neglect mentioning my good friends, Nathan Basik, and his wife, Irma Verónica Alarcón, who helped me with proofreading and linguistics, respectively (let alone various occasions for discussion over the dinner table at their home); and the fellow researchers at CRCC: Jim Marshall, Hamid Ekbia, and the "younger generation" folks: David Landy, Francisco Lara-Dammer, Abhijit Mahabal, Jay Muller, Eric Nichols, Michael Roberts, and Damien Sullivan.

Finally, no part of my work would have been completed smoothly without Helga Keller, the heart and soul of CRCC. Our dear research center could not function properly — if it could function at all — if Helga were not there to help all of us with our essential administrative prerequisites.

Harry Foundalis,
Bloomington, May 2006.

ABSTRACT

Phaeaco is a cognitive architecture for visual pattern recognition that starts at the ground level of receiving pixels as input, and works its way through creating abstract representations of geometric figures formed by those pixels. Phaeaco can tell how similar such figures are by using a psychologically plausible metric to compute a difference value among representations, and use that value to group figures together, if possible. Groups of figures are represented by statistical attributes (average, standard deviation, and other statistics), and serve as the basis for a formed and thereafter learned concept (e.g., triangle), stored in long-term memory. Phaeaco focuses on the Bongard problems, a set of puzzles in visual categorization, and applies its cognitive principles in its efforts to solve them, faring nearly as well as humans in the puzzles it manages to solve.

TABLE OF CONTENTS

LIST OF FIGURES

Part I: Bongard's World

CHAPTER ONE

Prologue with a Book's Epilogue

1.1 Introduction

In 1967, *Проблема Узнавания* (The Problem of Recognition) was published in Moscow. Its author, Mikhail Moiseevitch Bongard, then entirely unknown in the scientific community of Western Europe and the U.S.A., was interested in the automation of visual perception. He could afford only as much complication in the visual input of his domain as the limited abilities of computing systems of his time and place would allow. Thus, he opted for a black-and-white world of static figures in two dimensions, in which objects could be either outlined or filled, essentially with a single "color" used for both outlining and filling, and everything else considered "background".

Bongard's book was translated into English and published in the United States under the title *Pattern Recognition* (Bongard, 1970, pp. 656-661). A few years later, this translation caught the attention of Douglas R. Hofstadter, who was working on his book *Gödel, Escher, Bach: an Eternal Golden Braid* (henceforth GEB) (Hofstadter, 1979). Hofstadter, however, was less impressed by the ideas developed in the main text of the book than by an appendix that contained 100 problems of visual pattern recognition and categorization. He featured several of these problems in Chapter XIX of GEB, in a section titled "Bongard Problems" (henceforth BP's), which is the term that has prevailed thereafter in the literature.

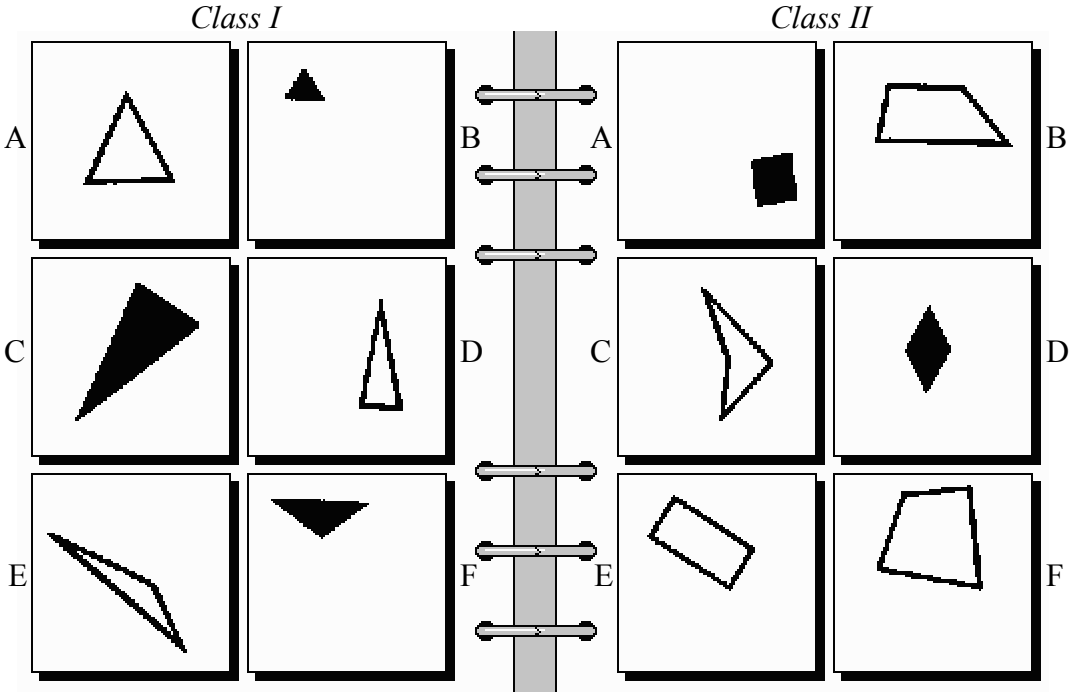


Figure 1.1: BP #6; one of the simplest Bongard problems, contrasting two patterns

A typical BP consists of 12 *boxes,*[1] six on the left and six on the right side, each of which contains a drawing.[2] The task of the BP solver is to discover a statement that describes the contents of each box on the left side, but does not describe the contents of any box on the right side. For example, in BP #6 (Figure 1.1) the description is simply "triangle". Quite often (and this is true for the more aesthetically pleasing BP's), there is a contrasting description that holds on the right side but not on the left, as is the case in BP #6 ("quadrilateral"). In what follows, solutions with such dual descriptions will be given with both parts (e.g.,

[1] Throughout the rest of this text, boxes will be indexed as shown in Figure 1.1: I-A, I-B, I-C, I-D, I-E, and I-F on the left side, and II-A, II-B, II-C, II-D, II-E, and II-F on the right side.

[2] Although all drawings shown in this text will be black-and-white only, this is not a necessary restriction imposed upon BP's. Indeed, as will be discussed in §2.3.1, one can easily imagine drawings including colors (e.g., true photographs), a third dimension, and even motion. Furthermore, there is no requirement that the number of boxes per side has to be six. As we will see later, at least one researcher (Maksimov, 1975) experimented with different numbers of boxes.

"triangle vs. quadrilateral"), but it must be noted that, at a minimum, the BP-solver is required to discover a description for the left side.

Hofstadter proposed in GEB a prolegomena to any future system that would attempt to automate the process of BP-solving, reducing his ideas to a number of principles that have heavily influenced the present work. He also created 56 additional problems, in which he pushed the ideas beyond simple pattern recognition to abstract relations and analogy-making (Hofstadter, 1977). The present author, who has also engaged in BP-creation, selected 44 of his own most attractive BP's to bring the total number of BP's to 200 (Foundalis, 1999), all of which are given in Appendix A.

Phaeaco (pronounced fee-AH-ko; see Appendix C for the origin of this name) is a cognitive architecture that focuses on BP's and proposes a computational approach for solving them. The same name also refers to an implemented system that, at present, demonstrates its competence in the automation of cognition by solving a number of BP's — most notably a subset of the 100 that were originally designed by Bongard.

1.2 A virtual tour in Bongard's world

Bongard's collection of problems has certain characteristics. An initial small number of them, through their solutions, exemplify some of the most basic characteristics of the domain, typical of what the BP-solver must be prepared to expect. Perceptual features such as "outlined", "filled", "large", "small", "concave", "curved", "vertical", "horizontal", "elongated", "open", "closed", and many more, all pop up in the solutions of the first few problems. Simultaneously, a number of elementary relations are introduced, including "more", "less", "above", "below", "up", "down", "leftward", "rightward", "equal", "non-equal",

and more. In what follows, the main characteristics of the domain of BP's are introduced, by selecting suitable representative problems for each case.

1.2.1 Solutions with difference in a feature value

Most features of shapes in the domain of BP's are continuous, but some of them are discrete. For example, Figure 1.2 shows a problem with a solution based on a feature that takes on the discrete values "outlined" and "filled". This feature will be called *texture*.[3]

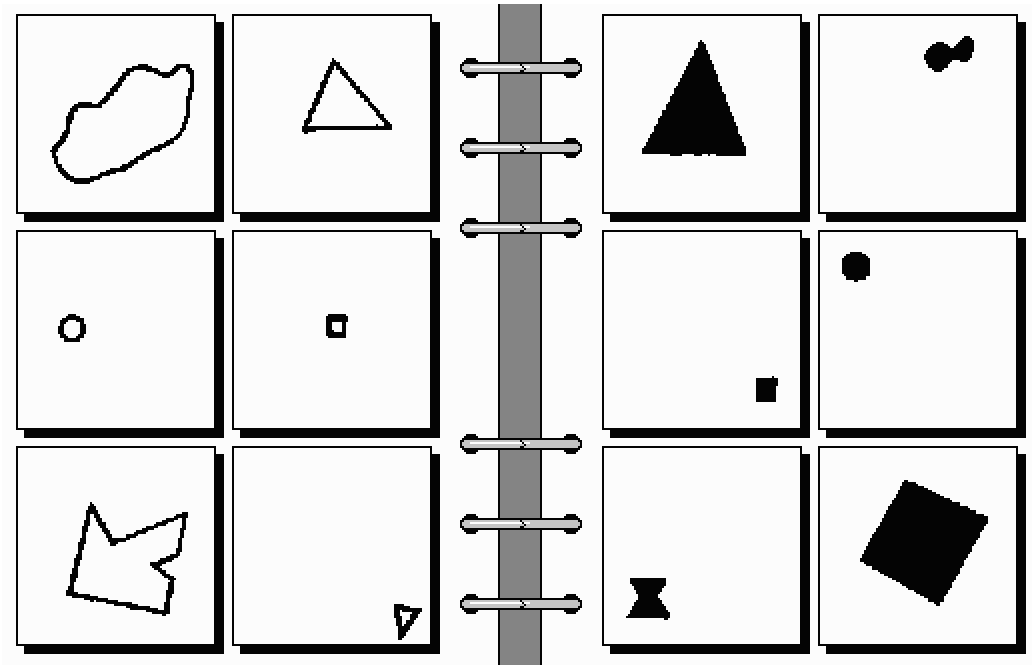


Figure 1.2: BP #3, exemplifying a feature with discrete values

BP #3 in Figure 1.2 belongs to a special category of a small number of problems that a person can solve "instantly" merely by contrasting the "colors" of the two sides (here, white vs. black), and seeing the groups that the corresponding

[3] Not to be confused with the *texture of a surface,* a term used in traditional image-processing.

shapes form (Treisman, 1986) — an issue that will be examined further in §11.1.1. Other kinds of features, however, can have a continuous range of values, and are not necessarily solved "instantly".

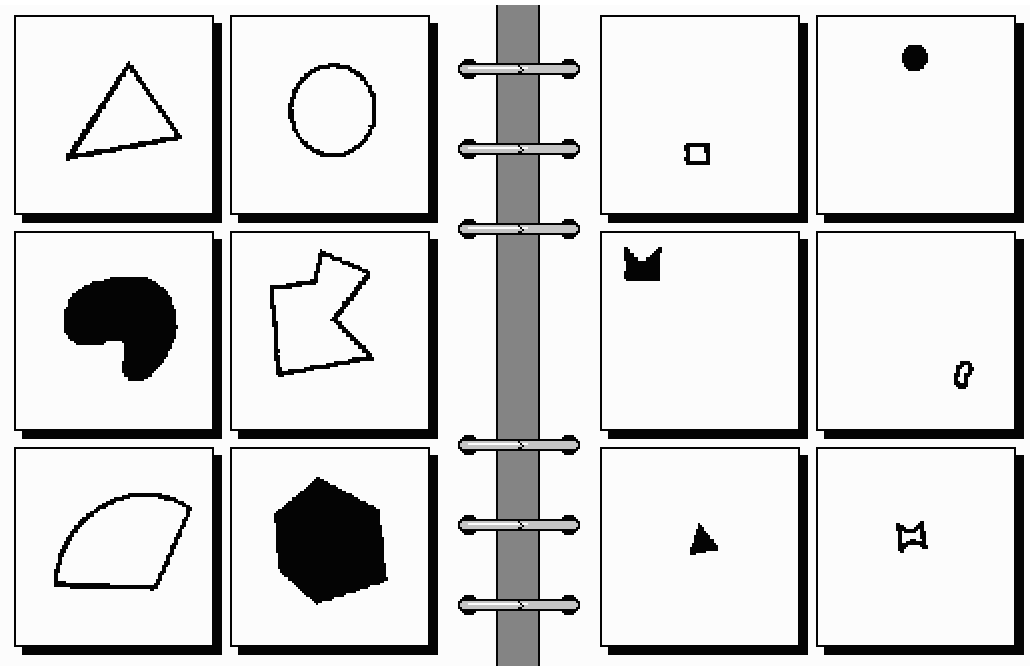


Figure 1.3: BP #2, exemplifying a feature with continuous values

For example, BP #2 in Figure 1.3 has a solution ("large vs. small") based on a feature that takes on continuous values. Whether some object is "large" is not an absolute characteristic, but a relative (contextual) one, based on the sizes of objects in the *other* group. Thus, the feature with a continuous range of values in this problem is the "area of an object".

A third subcategory of problems with solutions based on a difference in feature values employs the idea of *numerosity,* i.e., the number of something. The "something" can be anything countable: lines, objects, straight sides, vertices, curves, branches, and even abstract relations, such as "levels of nested-ness".

Although computationally the "number of something" can take on any value (such as, "1263 pixels"), the cognitive notion of numerosity is different, in that its resolution is finer at small values (1, 2, 3,…), but becomes progressively coarser for higher values (e.g., 47 is virtually indistinguishable from 48); more will be explained in §7.3.

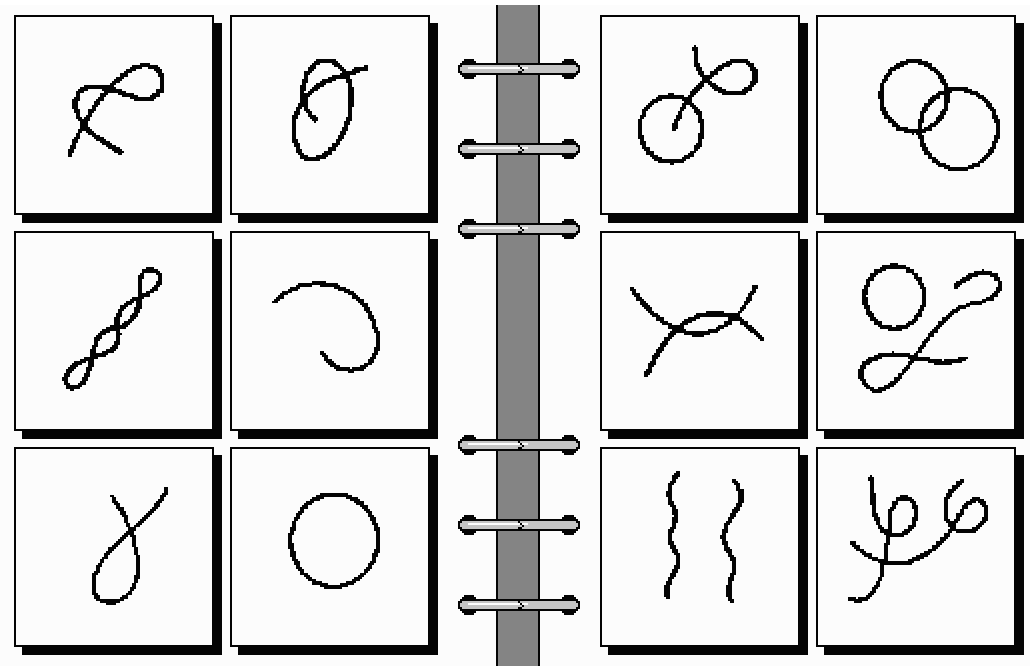


Figure 1.4: BP #31, a solution based on numerosity (of curves)

Figure 1.4 shows BP #31, the solution of which is "one curve vs. two curves". In BP's, the numerosity values employed in a solution are never in the range in which human cognition has a hard time (e.g., "47 vs. 48").

1.2.2 Existence

A second common theme among the solutions of BP's is the existence of something. The next two BP's exemplify this idea.

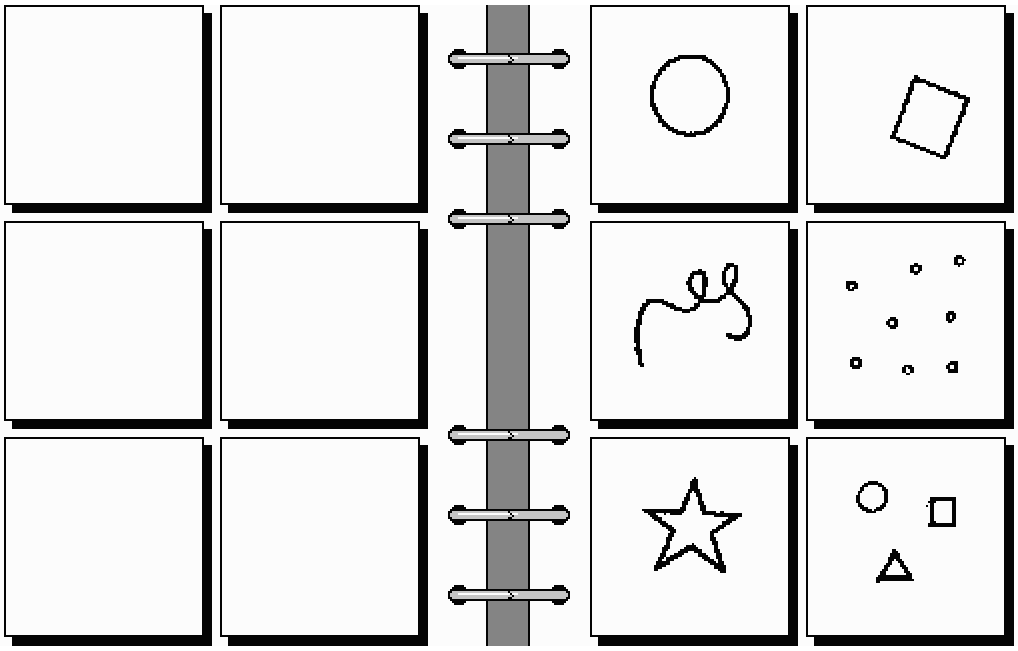


Figure 1.5: BP #1, existence of objects (on the right side)

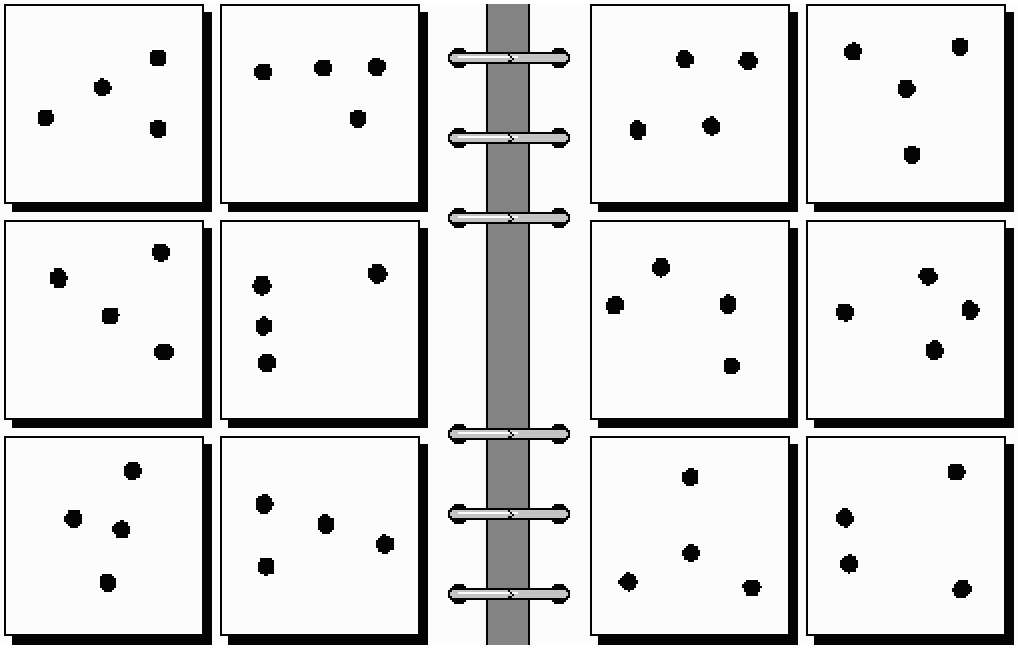


Figure 1.6: BP #40, existence of imaginary straight lines

BP #1 (Figure 1.5) is more interesting than a casual look at it might reveal, in that its solution can be arrived at in two ways: one is the "immediate" way, by contrasting the "whiteness" on the left side against the existence of dark areas and lines on the right (similar to BP #3, Figure 1.2); but another way is to contrast the existence of an internal representation on the right side (a thought of "something that exists", however vague that might sound now — it will become clearer in later chapters) against the nonexistence of a representation on the left. This second way of reaching the solution works at a meta-level, because it describes *representations* of objects, rather than the visual objects themselves.

1.2.3 Structural and relational differences

The term "structural difference" refers to visual objects forming structurally different patterns, as in BP #6 (Figure 1.1). Another example is presented below.

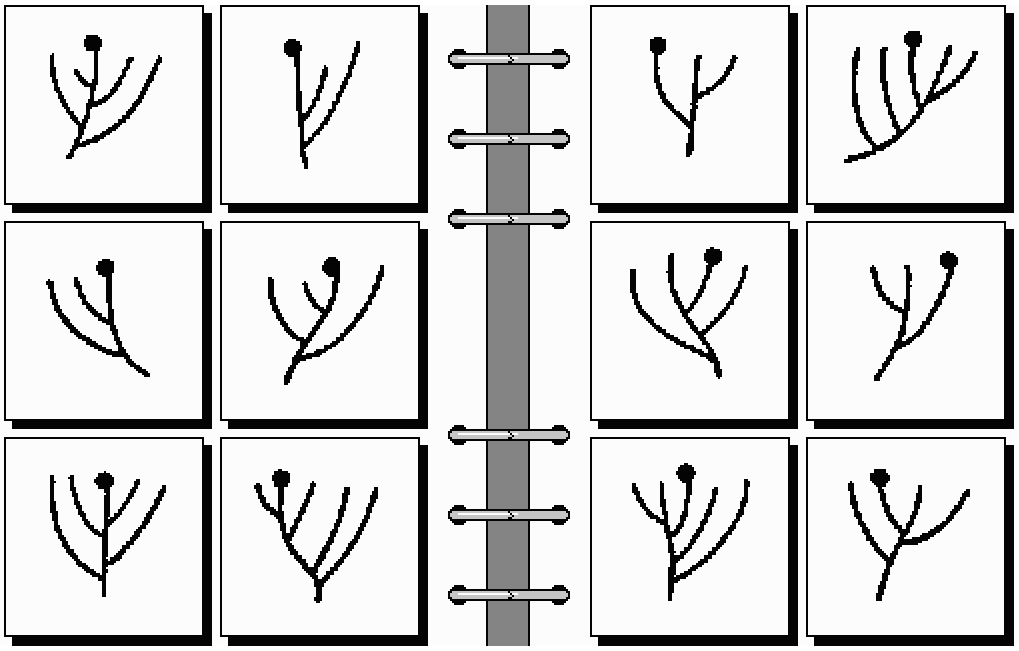


Figure 1.7: BP #69, exemplifying a structural difference

BP #69 (Figure 1.7) has a solution with a structural difference: the pattern on the left side is, "a tree with a dot at the tip of the trunk", whereas the pattern on the right is, "a tree with a dot at the tip of a branch".

Slightly different are the rules that include a "relational difference".

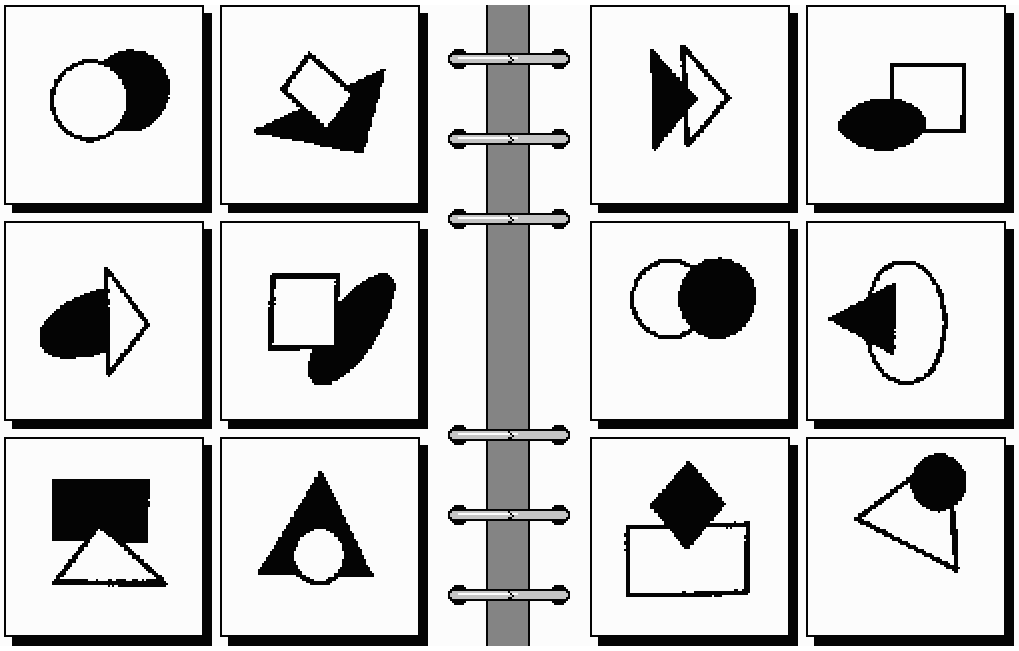


Figure 1.8: BP #45, exemplifying a relational difference

The rule in BP #45 (Figure 1.8) is based on a *relation*: on the left, the outlined figure is *on top of* the filled one, while on the right the relation is reversed.

It now becomes clear that it is not always possible to make a sharp distinction among structural and relational differences. On the one hand, the objects in the boxes of the left side of BP #45, for example, might be seen as forming a *structure* consisting of two parts, one filled and the other outlined, such that the outlined part always occludes the filled part. On the other hand, the tree-like structures in Figure 1.7 might be seen as described by a *relation*: the dot stands *on*

the tip of the trunk on the left, but does not do so on the right. Thus, we reach a notion that will become very familiar in this work:

A *visual pattern* is an abstraction, or generalization, that is based on a number of concrete visual objects. Examples are the triangles and quadrilaterals of BP #6 (Figure 1.1), the tree-like objects of BP #69 (Figure 1.7), and relations such as those of BP #45 (Figure 1.8). A visual pattern is a sort of *statistical average,* in which the parts of a structure, or relation, and the values of features, are averaged. This notion is of crucial importance in the architecture of Phaeaco, and will be described more rigorously in chapter 8.

1.2.4 Relation within a single box

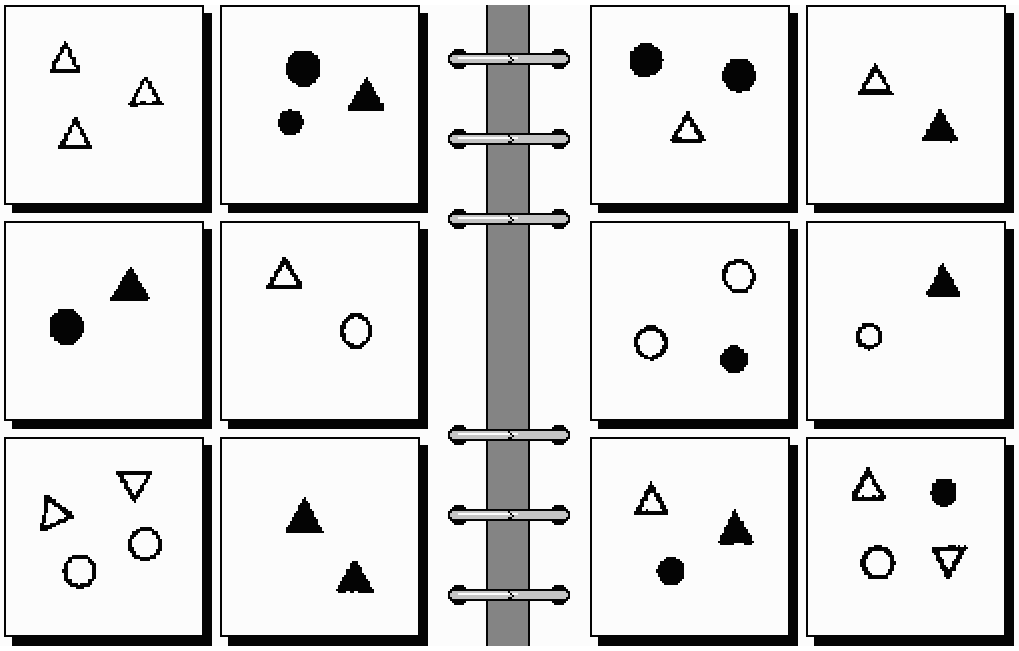


Figure 1.9: BP #56, with sameness within each single box

BP #56 (Figure 1.9) is of yet a different kind. Here, the relation on the left side is "uniformity of texture (§1.2.1)", but the relation (uniformity) is to be found not

across boxes, but within each single box. Thus, the solution for BP #56 is, "every object has the same texture". This type of solution generalizes into one that reads, "relation X holds on feature Y". (See also BP's #22 and #173 in Appendix A.)

1.2.5 Solutions and aesthetics

There are certain BP's, often very interesting ones, that do not easily lend themselves into the previous classifications. Consider Figure 1.10.

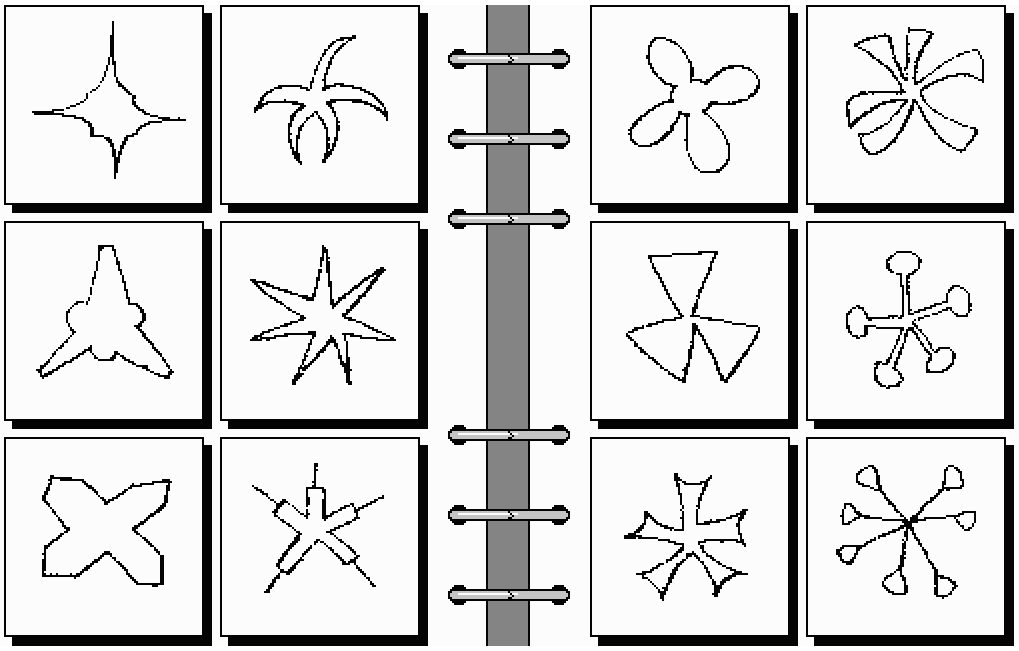


Figure 1.10: BP #108, types of "flowers"

BP #108 (Hofstadter, 1977) contains "flowers" in all 12 boxes, but the "petals" of the flowers on the left taper off, while those on the right grow thicker, in some abstract sense. This problem could be categorized as "different structures, or patterns" (§1.2.3), but what exactly it is that makes these structures different (if we want to be precise and avoid references to terms such as flowers, petals, etc., which are non-geometrical and foreign to a non-terrestrial culture) is hard to

define in formal terms, using geometric primitives. The simplification that occurs by using terms such as "flower" and "petal" is what makes this BP appealing.

A totally different example is shown in the following figure.

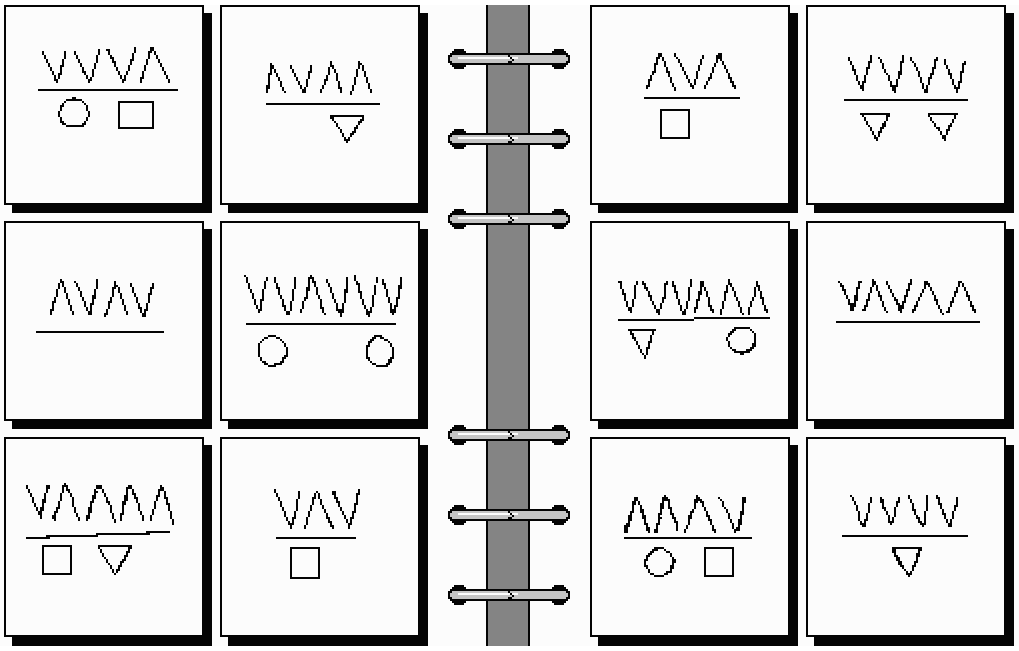


Figure 1.11: BP #121, with a code-breaking type of rule

In BP #121 in Figure 1.11 (Hofstadter, 1977), the shapes above and below the horizontal line in each box adhere to a kind of *code,* which, on the left, can be stated as follows:

- Each ⋁⋁ corresponds to a circle
- Each ⋀⋀ corresponds to a triangle
- Each ⋁⋀ corresponds to a square
- Each ⋀⋁ corresponds to "empty"
- Finally, left-over (single) ⋁'s or ⋀'s are ignored

The assignment of ⋁'s and ⋀'s to shapes on the right side is exactly reversed.

Other aesthetically pleasing problems are those with solutions having multiple levels of description, or "recursion". For example:

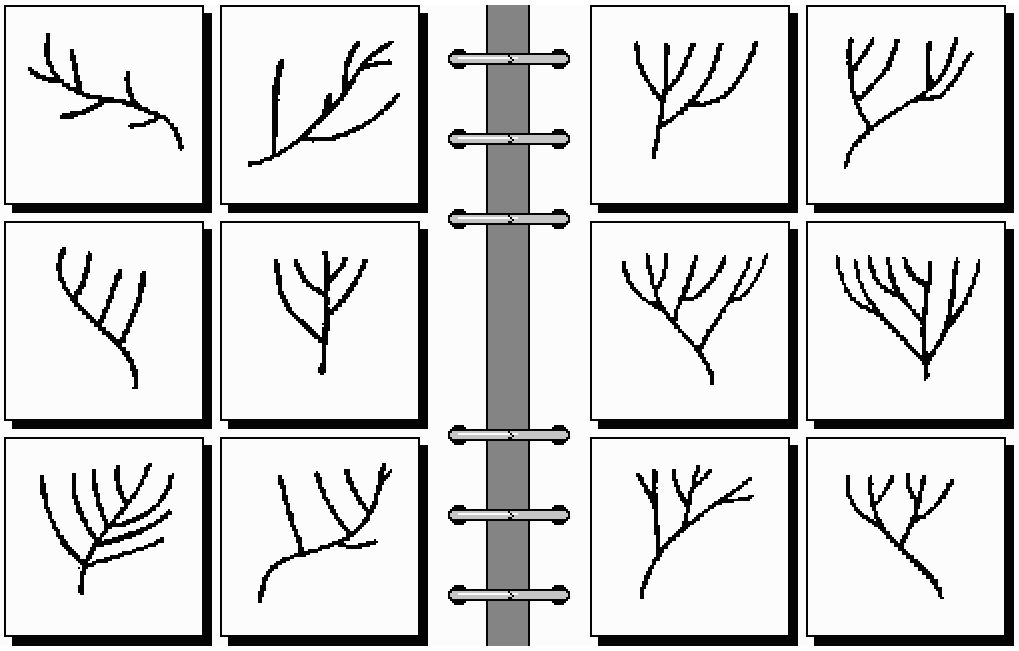


Figure 1.12: BP #70, one vs. two levels of description

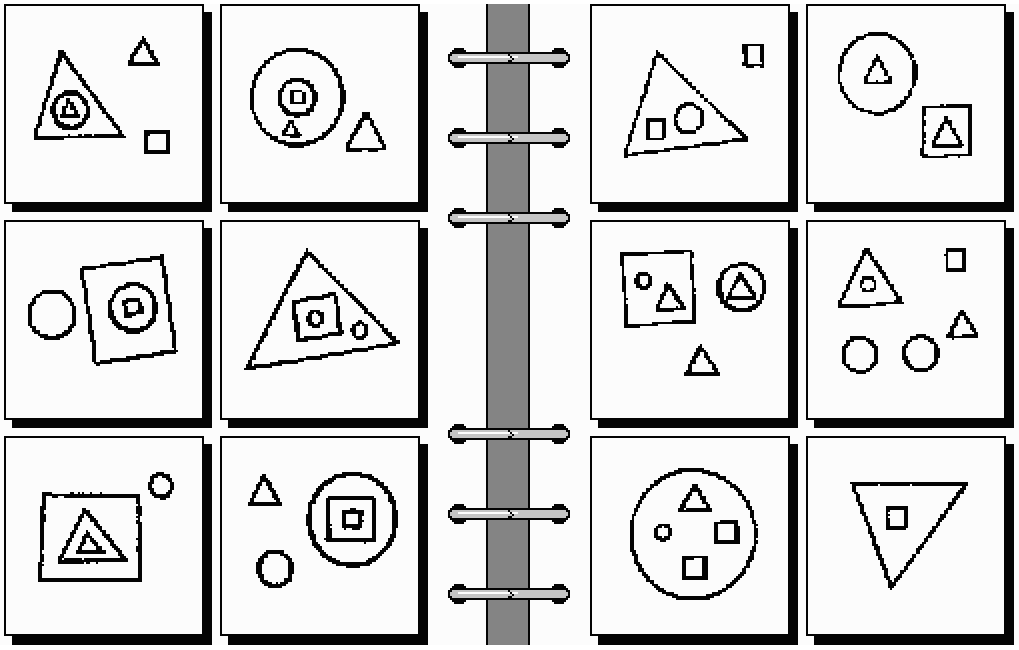


Figure 1.13: BP #71, two vs. one level of description

In BP #70 (Figure 1.12), the dual solution is: "major stem branches into single twigs, versus major stem branches into twigs that may branch for a second time". In BP #71 (Figure 1.13), which looks superficially very different, the theme is essentially the same (with the roles of the sides reversed) at a more abstract level: "object within object within object, vs. object within object". In other words, two levels of recursively using the relation, versus a single one.

It is not coincidental that in Bongard's list of problems, BP #71 appears immediately after its twin sibling, BP #70. Bongard made use of the idea of *priming* in his collection. Often, after introducing an idea through a problem, he uses it — but disguised behind different façades — in subsequent problems.

Last but not least, not all BP's are relatively easy to solve: some of them are downright hard.

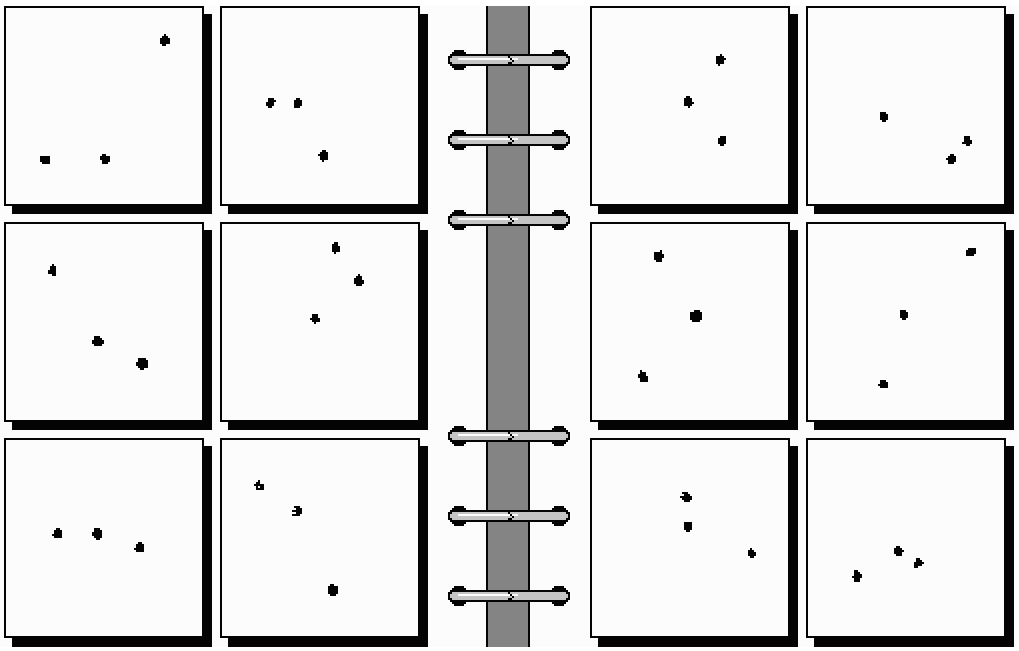


Figure 1.14: BP #112, a paragon of simplicity

BP #112 (Figure 1.14), for example, looks deceptively simple (Hofstadter, 1977). Yet it is probably one of the hardest problems in the entire collection of 200. Another hard problem, BP #180 (Foundalis, 1999), is presented below. (The solutions of BP #112 and BP #180 are left as challenges for the reader, but can be found in Appendix A as a last resort.)

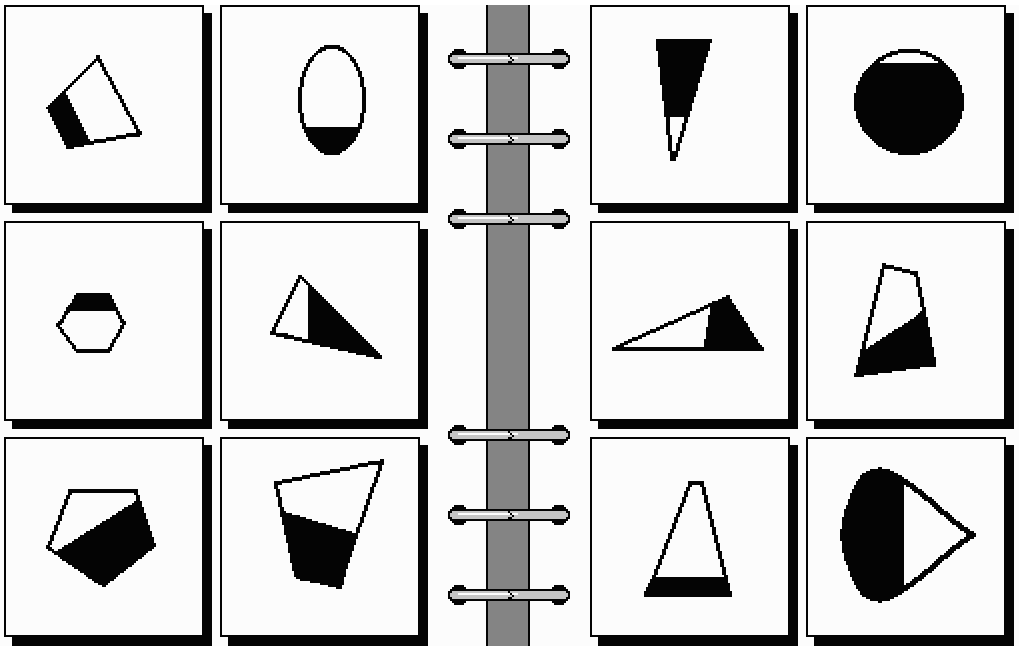


Figure 1.15: BP #180, another seemingly simple but hard problem

There are many more ideas that have been expressed in BP's (some of which will be discussed in subsequent chapters), and even more ideas that can be *potentially* expressed. Bongard's domain is profoundly rich in cognitive content — much richer than hinted at in the present introduction. In the chapters that follow, an attempt is made to reveal this complexity, and the means by which Phaeaco navigates the challenges.

CHAPTER TWO

Why Are BP's Cognitively Interesting?

2.1 Questioning microdomains

At least since the early 1970's, a dilemma has been facing Artificial Intelligence (AI). In the early days of AI, systems were often built by choosing a problem domain and then stripping away its real-world "burden" — the details that grounded the problem in the real world, and which were considered superficial. Thus purified and crystallized, "the problem" — evidently a caricature of its real-world counterpart — was modeled in a computing system. The modeler's claim would be, typically, "The problem has been solved in its abstract form; it now suffices merely to add some details to turn it into an implementation of the original problem in the real world."

The trouble with this approach is that the missing details, which were originally considered superfluous, are in reality the determining factor that renders the crystallized solution useless. This discovery is known as "the scaling-up problem" in AI: it is the thesis that it is impossible to start from the "crystal" and work one's way up toward the real-world problem by merely adding details in a piecemeal fashion, thereby obtaining an intelligent computer program. Typical examples of this situation can be found in programs that worked in so-called "blocks worlds" (i.e., simulated geometric solid blocks placed on a tabletop) (Huffman, 1971; Waltz, 1975; Winograd, 1972; Winston, 1975).

A different problem concerns the motivation and goals of early work in AI. Researchers would typically select a microdomain, not in order to investigate the foundations of cognition, but to be able to claim that the microdomain is "solved", or "automated" by their approach, while offering no overall perspective on how to generalize from their particular case. Some examples are various expert systems that were proposed in the 1980's as computerized solutions for engineering and medical decision-making issues, and Thomas Evans's ANALOGY program, solving geometric analogy problems taken from IQ tests (Evans, 1968). Figure 2.1 shows an example of a problem that could be solved by Evans's program, which, contrary to widespread belief, did not examine images at the pixel level. Instead, its input comprised hand-prepared relational expressions written in a Lisp-like form, representing the images to some extent. For the pitfalls of this approach, see also §4.1.

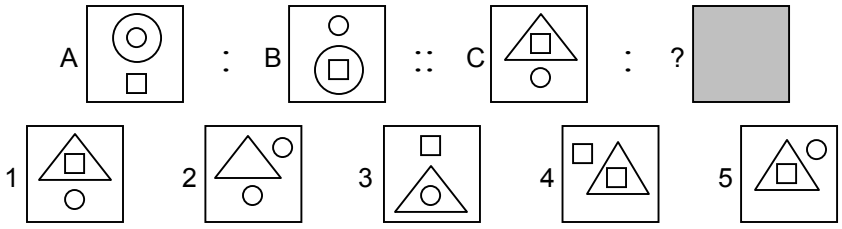


Figure 2.1: One of Evans's geometric analogy problems: A is to B as C is to ? (answer: 3)

Given this background in AI, one might naturally wonder whether automating the process of BP-solving is merely another instance of domain trivialization. Indeed, one might imagine that the real-world domain in this case is human vision, the problem at hand is how to automate it, and that Phaeaco trivializes human vision by reducing it to Bongard's microworld.

The short response to this concern is that Phaeaco's goal is *not* to provide a model for successfully automating visual perception or image processing. Instead,

Phaeaco uses the domain of BP's as a litmus test for its proposed set of principles that presumably lie at the core of cognition. Phaeaco should be construed as a cognitive architecture rather than as an automated BP-solver.

A longer response is provided in the remainder of this chapter. First, it is demonstrated that the domain of BP's includes some elements that appear to be quintessential in human cognition. And second, it is argued that the domain itself is deceptively perceived as a microdomain, and should not be understood as being limited by rigid boundaries. In the domain of BP's the mind is the limit, as will hopefully become evident in the sections that follow.

2.2 BP's as indicators of cognition

The following sections examine the relevance of Bongard's domain to cognition. In each subsection, some fundamental aspect of cognition is examined and shown to be required for the solution of some BP that has certain properties.

2.2.1 Pattern formation and abstraction

Suppose that an observer who has no prior knowledge of alphabets, letters, or other culturally related notions is given the following instances of visual input one at a time, each appearing shortly after the previous one is erased (Figure 2.2).

Figure 2.2: Instances of visual input

An observer with human-like cognition neither forgets each instance after it is replaced by its successor, nor simply stores it in memory; instead, the observer

forms a *visual pattern* out of the given input instances. This pattern can be either a sort of "summary representation" of the individual instances, as in the prototype theory of concepts (Hampton, 1979; Rosch and Mervis, 1975; Smith and Medin, 1981), or the individual instances themselves, as in the exemplar theory of concepts (Lamberts, 1995; Medin and Schaffer, 1978; Nosofsky and Palmeri, 1997), or some alternative (Goldstone and Kersten, 2003; Hofstadter, 1995a; Murphy and Medin, 1985; Rehling, 2001). (More on theories of conceptual representation in §6.1.) Regardless of what components constitute the visual pattern, the latter encapsulates some information. For example, the observer knows that the slopes of the slanted lines in Figure 2.2 are within some bounds, and the altitude of the horizontal-like line is typically within some limits. Not only are there limits to the variation of such features, but there is also a mean value and a variance. These are perceptible because the observer can reproduce a "typical" instance upon request, which will have the horizontal-like line, for example, positioned close to the middle of the figure. Conversely, the farther away from the middle this line is placed, the more atypical the produced instance will look.

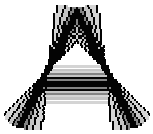


Figure 2.3: One way to depict a summary representation of the input in Figure 2.2

Figure 2.3 shows pictorially some of the statistics perceived by the observer. In particular it shows the average and variation in slope of the two slanted sides, the location of the horizontal line, and possibly the width of various lines (but note that lengths are not drawn to scale). It also omits several other features that are possibly perceived, such as the occasional short lines at the bottom and top of

some input instances (the serifs). It must be emphasized that Figure 2.3 depicts neither a visual pattern according to some particular theory of conceptual organization (e.g., the prototype theory), nor Phaeaco's representation of a pattern. It merely suggests the idea of a "visual pattern" for the purposes of the present discussion.

Visual patterns are formed by the contents of the six boxes on each side of a BP. Often, the patterns of the two sides do not exhibit an immediate difference, i.e., the observer perceives *the same* pattern on both sides after just a cursory look.

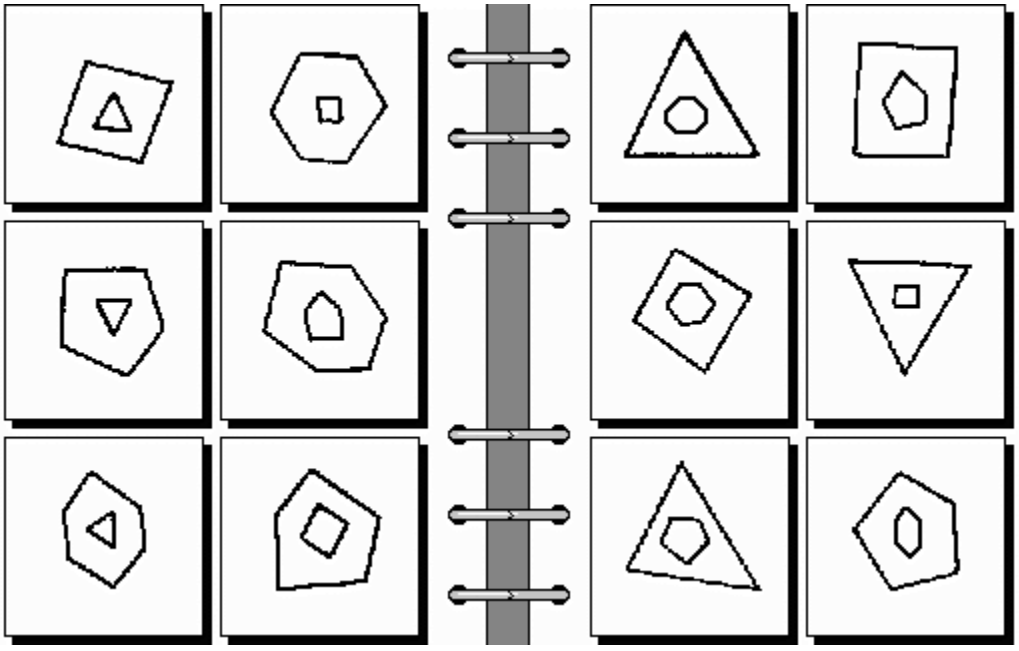


Figure 2.4: A cursory look allows perception of a single pattern only in BP #53

Consider BP #53 in Figure 2.4. The pattern formed on both sides can be described succinctly as "polygon inside polygon". Most solvers discover the difference in the number of sides between the inside and outside polygons only after a careful examination of each individual box and after reaching the idea

"count the sides". Another such example is BP #108 (Figure 1.10). Some problems, however, the solutions of which are usually based on a structural difference (§1.2.3), do exhibit patterns on the two sides that are more readily perceptible as different, as in BP #6 (Figure 1.1) and BP #183 (Figure 2.5).

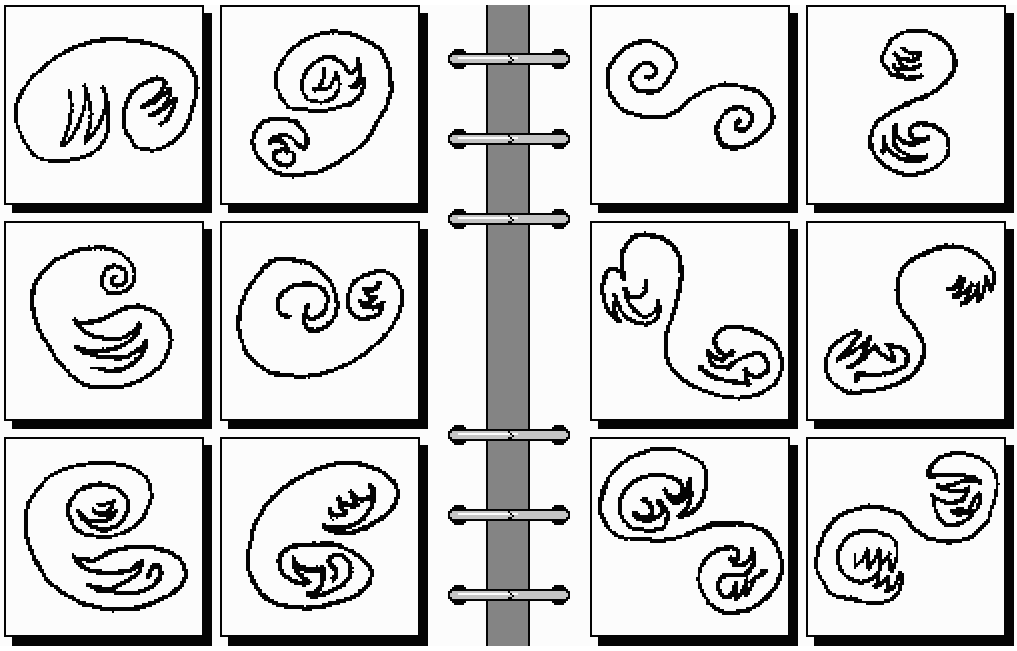


Figure 2.5: The different patterns in BP #183 are readily perceptible

Nonetheless, a visual pattern can be formed whether or not the input instances are similar. Consider BP #1 (Figure 1.5): the solver, observing the right side, forms a pattern that can be described as "some figures". Naturally, if the domain provides a richer context (e.g., real-world, animated photographs), the pattern on the right side of BP #1 will be more specific (e.g., "black-and-white, two-dimensional, still figures"). The way the context influences what is and is not to be included in a pattern is subtle and fundamental in cognition, and will be examined in more detail in §8.2.3.

2.2.2 Pattern-matching, recognition, and analogy-making

Just as storage is meaningless without retrieval, so pattern formation is useless to a cognitive system without *pattern-matching,* i.e., the ability to retrieve a pattern that matches best with a new input instance. Bongard and other scientists have long appreciated the importance of this cognitive ability. Hofstadter has opted for the term *analogy-making* (Hofstadter, 1995a), because he is interested in the more abstract, higher-level, fluid, human-specific manifestation of this mechanism. In truth, there is no sharp dividing line between "pattern-matching" and what most cognitive scientists would accept as an instance of analogy-making. Consider Figure 2.6.

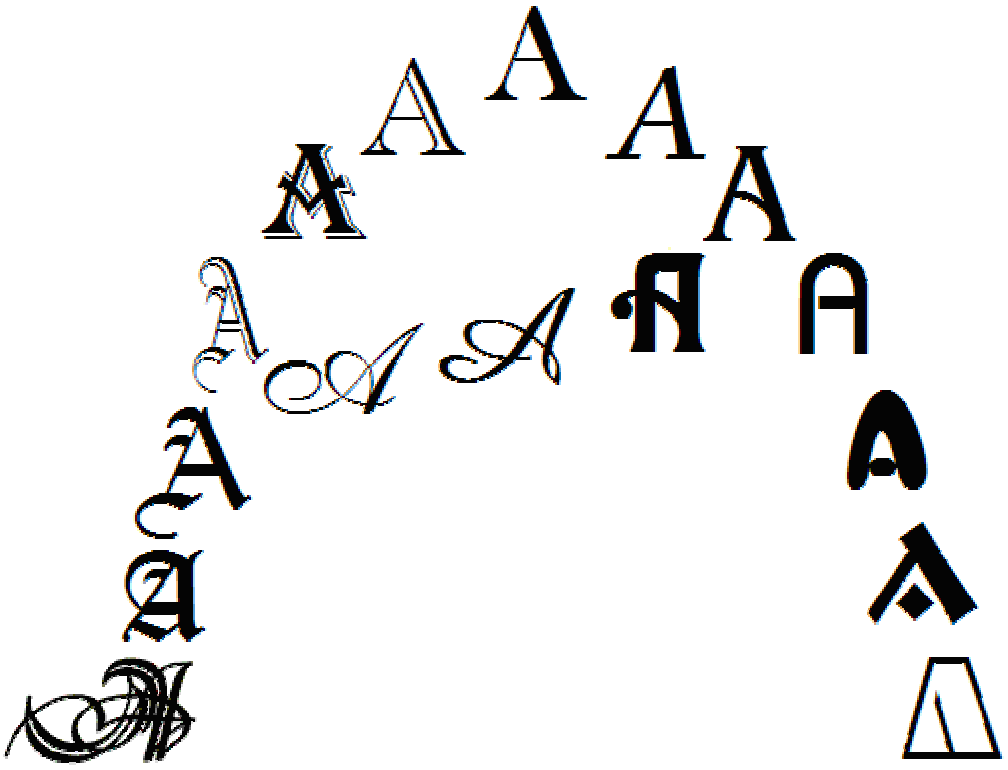


Figure 2.6: Input instances that progressively deviate from a given instance (top)

The learned pattern depicted in Figure 2.3 *matches* (i.e., the statistics of its features include squarely the features of) the top instance of Figure 2.6. Moving progressively, however, toward the two instances at the bottom, we pass into an area where a mechanism finding any resemblance between the learned pattern and these two instances would be more appropriately called "analogy-making". This is because the statistics of the learned pattern describe neither the complexity of the bottom-left instance, nor the simplicity of the bottom-right instance; hence, neither of these two instances matches the pattern. Still, some components of their structures can be seen as *analogous* to components of the pattern. Somewhere between them is a gray area of input instances for which it is difficult to call the mechanism either "pattern-matching" or "analogy-making" (see §8.4).

The domain of BP's abounds with problems throughout the entire spectrum from pattern-matching to analogy-making.

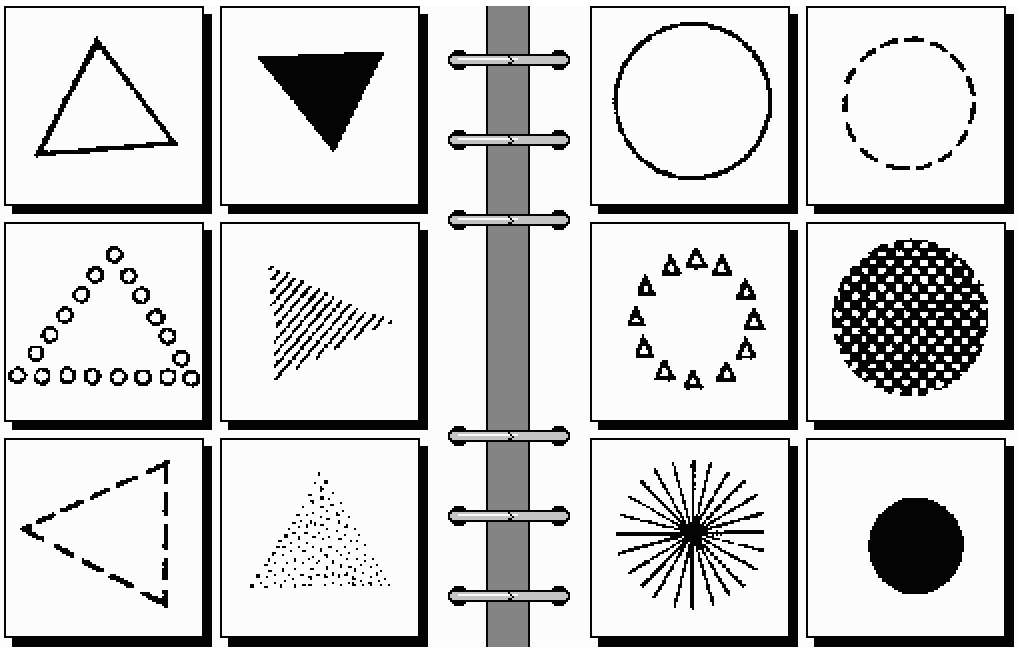


Figure 2.7: Analogy making in BP #97 (or is it pattern formation and matching?)

BP #6 (Figure 1.1, "triangle vs. quadrilateral") is an example of simple pattern formation and matching: the pattern of a triangle is formed out of the six instances on the left, and is contrasted with the pattern of a quadrilateral formed out of the six instances on the right. Similarly, the pattern of a triangle is at work on the left side of BP #97 (Figure 2.7), but in this case the degree of abstraction required is higher: the solver must perceive, for example, that each row of small circles in box I-C forms (is *analogous* to) a side of a triangle — a perceptual feat that is neither automatic, nor trivial, as will be shown in §3.1.1. BP #170 (Figure 2.8) moves even further along the spectrum toward analogy-making, requiring the solver to retain only the essence of the structure in each of the 12 boxes, mentally eliminating the irrelevant details. Seeing the analogous parts in these figures is an example of analogy-making (see also §8.4).

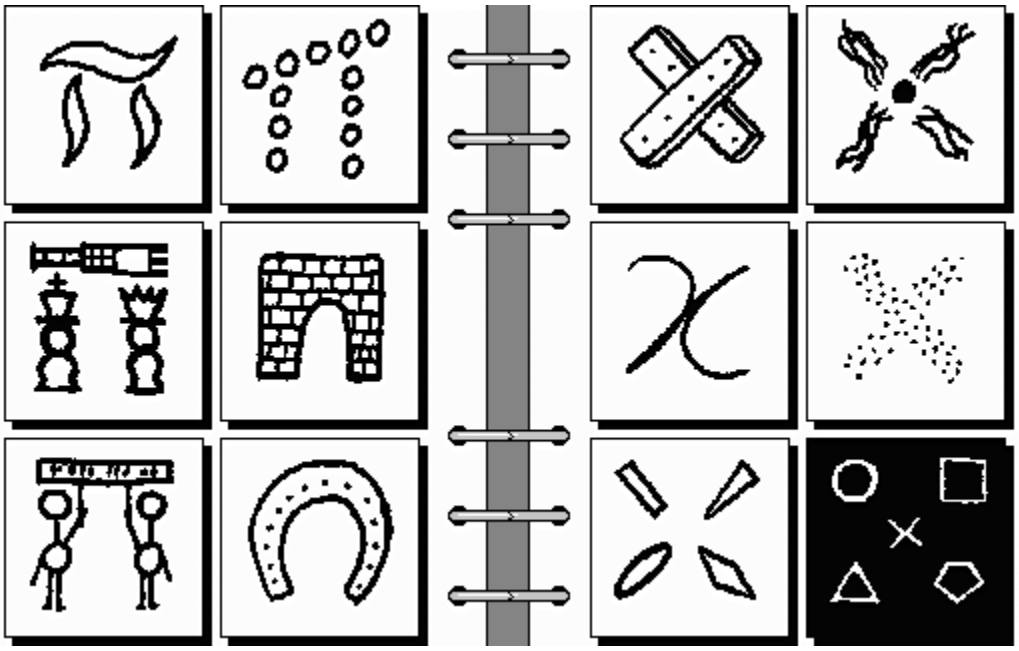


Figure 2.8: Analogy-making seems to be at work on the left and right sides in BP #170

2.2.3 Clustering and Categorization

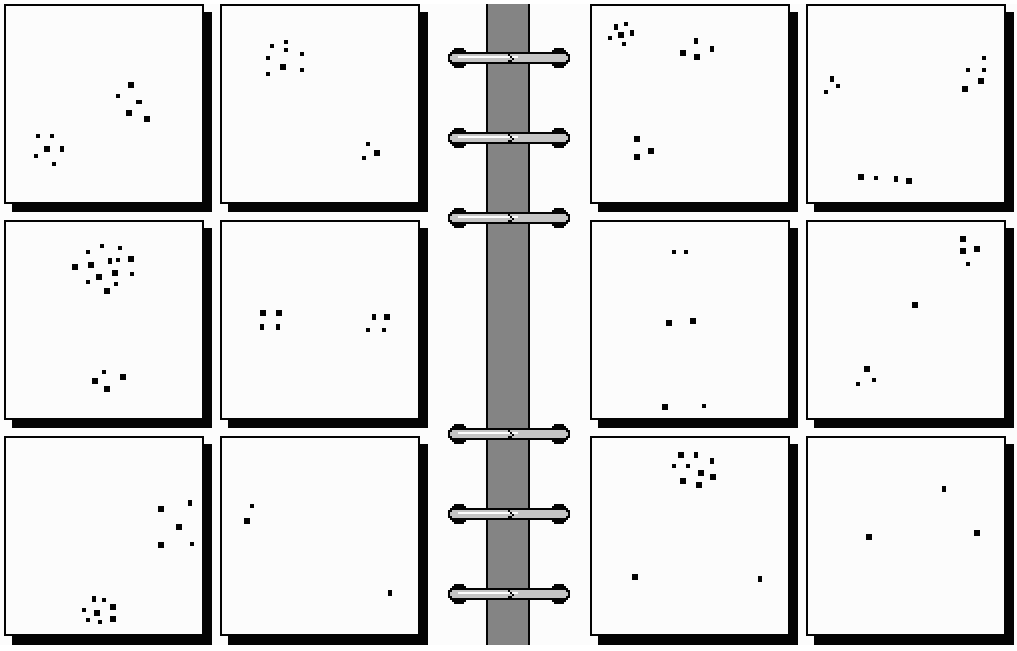


Figure 2.9: BP #166, an example of involuntary clustering

Consider BP #166 (Figure 2.9). Looking at this problem, we instantly see not merely dots, but clusters of dots. The construction of clusters of "things that go together" is another fundamental mechanism of cognition. It is the same mechanism that allows us to separate leaves from pebbles on the ground; place toasters, blenders, ovens, refrigerators, and coffee-makers in one mental category ("electrical appliances"); and even construct *ad hoc,* spontaneously manufactured concepts under pressure, such as children and jewelry as "things to take out of a house in the event of a fire" (Barsalou, 1983).

The placement of dots into groups in BP #166 is automatic: our visual system cannot avoid the perception of dot-clusters. In other cases, however, some mental effort is required to perceive the categories. Consider BP #90.

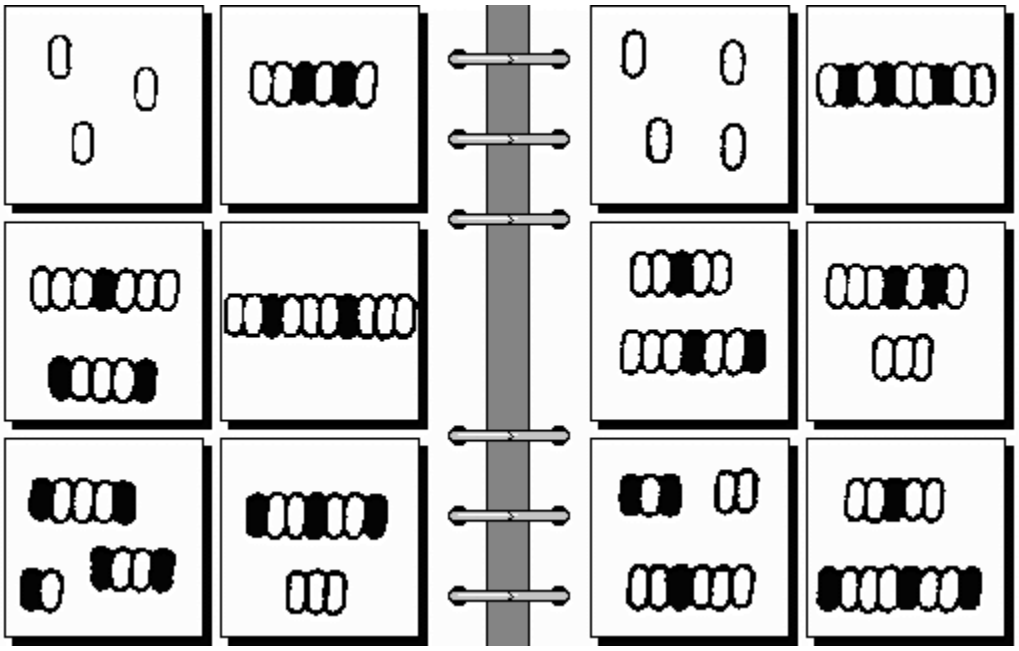


Figure 2.10: What constitutes a cluster in BP #90 is not immediately obvious

A cursory look at the ovals in BP #90 (Figure 2.10) might lead us to perceive first the clusters consisting of ovals that touch each other (i.e., the connected components in each box), but these are irrelevant to the solution. The solution can be reached only via the idea (conscious or subconscious) of focusing on white ovals connected with each other. This idea is reached under the (mild) pressure constituted by the thought "I have been asked to solve BP #90". A more urgent pressure, such as the smell of burnt material in a house, can lead to the previously mentioned *ad hoc* category of "precious object". Although the special term *goal-derived category* has been reserved for this latter kind of categorization in psychology, Phaeaco treats all categorization in a uniform way (see chapter 8).

It might be thought that categorization and pattern-formation (§2.2.1) are the same mechanism. To avoid any possible confusion, "categorization" will be used in the present text to refer to the assignment of an element to an already existing

cluster, whereas "pattern-formation" will refer to the process of clustering, i.e., the formation of a category, which, as will be explained in chapter 8, includes a core, a halo, and possesses statistical properties.

2.2.4 Memory and learning

The ability to learn new patterns, storing them in a long-term memory (LTM) and retrieving them later, is a fundamental property of cognition that can be examined in the domain of BP's. Forming a pattern (§2.2.1) is, of course, a type of learning; but it can occur also in short-term memory (STM), and represents a single facet of the complex problem of learning. STM-type learning is exhibited by simple[4] biological creatures (e.g., spiders) as *habituation*: a spider can be "habituated" to stop rushing toward the source of stimulation on its web after repeatedly finding out that the source is inedible (Arms and Camp, 1988). More complex creatures, however (e.g., most vertebrates, some mollusks, etc.), possess an LTM, which makes them capable of exhibiting more cognitively interesting behaviors, including (among a few mammals) the sense of "self", which they acquire by being able to know that they are "the same" individuals they were days, months, and years ago. A biological creature lacking LTM is more or less an automaton that responds to stimuli only on the basis of "now" (or, at best, "a few moments ago").

It is important that a BP-solving system possess LTM for the following reason. Consider a system *S′* that possesses LTM and a system *S* that lacks it, all other features of *S′* and *S* being identical. Assume that among the identical features shared by *S′* and *S* are a set of visual primitives *P*. Then *S* is capable of solving only those BP's with solutions that can be expressed by a relatively

[4] To avoid controversy regarding the biological meaning of "simple organism", the term is used here to mean an organism that appeared relatively early in evolutionary history.

simple combination of elements of *P,* whereas *S′,* being able to learn new concepts — turning them into a new layer of "primitives" *P′* in LTM — is in a position to reach more directly the solution of a much larger number of BP's. Although *in principle S* is also capable of reaching the solution of the same BP's as *S′* (since, after all, all solutions must be expressible by the same set of underlying primitives *P*), this remains a theoretical possibility only: a capable and careful tutor can lead *S′* into learning those concepts (*P′*) that are relevant to the solution of a large number of BP's, whereas *S* must struggle to "discover" those concepts by itself, always holding everything in LTM. Given the exponentially large number of concepts that are combinatorially expressible in terms of primitives, chances are that *S* will not reach the solution of those additional BP's within a reasonable amount of time.

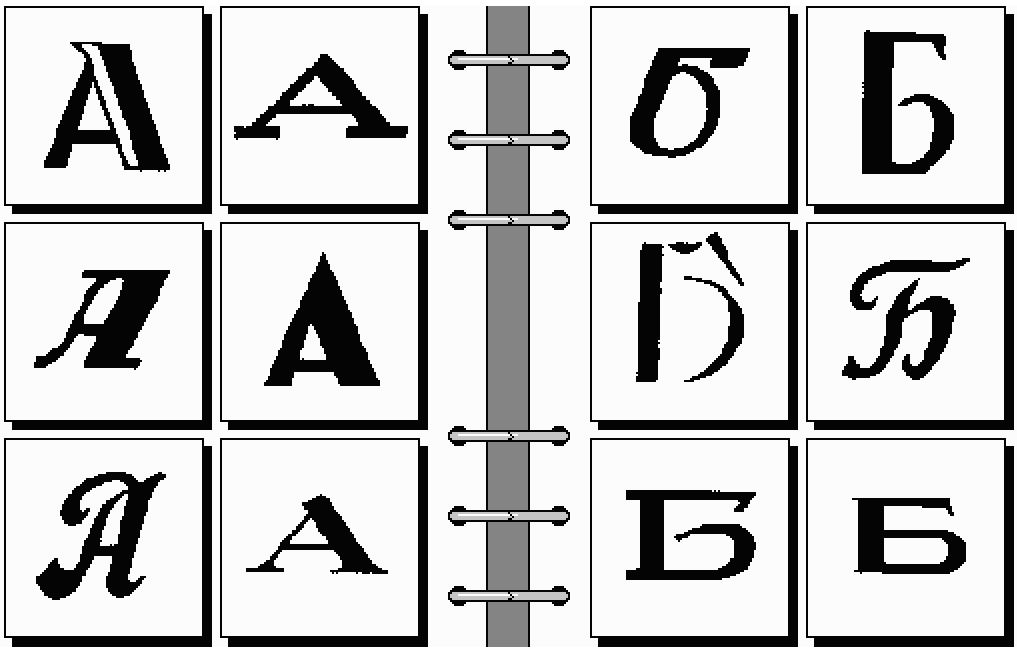


Figure 2.11: Learning and an LTM seems to be necessary in BP #100

A BP that demonstrates this idea is the last one in Bongard's original collection. The boxes on the left and right side in BP #100 (Figure 2.11) seem to be insufficient by themselves to define what each corresponding pattern is. Geometrical properties, such as a closed area, a pointy top, or two "legs" at the bottom, are inadequate for a definition of the pattern either because some members of the defined side lack those features, or some members of the other side possess them. Only a learner who has been trained by hundreds of examples of the Cyrillic letters "А" and "Б" in various fonts and has created such concepts in LTM is able to solve this problem.

One might counter-argue that the patterns for "А" and "Б" can be learned in STM alone, assuming each side of this BP contains hundreds of boxes. Although this is true, there are two problems with this idea: first, we (possessing LTM) are able to solve BP #100 *as it is,* with only six boxes per side; and second, a system that forgets all that it has learned once it is turned off and has to undergo an arduous process of re-learning by being exposed to hundreds of examples each time is utterly unsatisfying. No real cognitive creature blanks its memory and re-runs the education of its childhood every time its cognition is reactivated after a period of inactivity. Accordingly, Phaeaco possesses LTM, which resembles the Slipnet of Copycat (Mitchell, 1990), but has certain additional features. The details are discussed in chapter 9.

2.2.5 Design and creativity

An issue that exceeds the scope of the present thesis (and yet must be mentioned, because it suggests a direction for future development) is that of designing BP's. Can a program exhibit *creativity* by designing genuinely new BP's that not only present a challenge to the solver but are also aesthetically pleasing (§1.2.5)? How is creativity limited (if at all) by the number of primitive ("hardwired") perceptual

elements (i.e., innate knowledge of points, lines, slopes, etc.) of the program? Is Phaeaco's architecture sufficient to address the problem of creativity, or would a major architectural reorganization be necessary? These are all interesting questions that can be explored in future projects.

2.2.6 Language and communication

By addressing the *grounding problem* in language, the BP domain can be a platform for generating and testing hypotheses concerning relationships between vision and language. Phaeaco builds relatively rich internal representations of the figures it receives as input, but describes linguistically the attributes of such representations in a trivial way, by outputting pre-manufactured[5] English words that more or less correspond to such attributes (more in §5.2). A more thorough linguistic approach can be envisaged, in which it is learned that the *morphemes* of a language are formed out of smaller constituents (graphemes or phonemes), and that such linguistic units (the morphemes) are associated with ("grounded" in) visual percepts. Combinations of morphemes can then be used to describe more complex portions of representations. A proper approach in such a linguistic system requires that the system learn the syntactic rules of the language, according to which morphemes are combined together to express internal representations. It would then be possible to examine the problem of language development and communication between the system and a human interlocutor, which is absent from the present stage of Phaeaco's implementation.

[5] This means that the English words that Phaeaco outputs at present have been hardwired into its code and resource files, although as will be explained in §5.2 there is a mechanism by which new words can be added.

2.3 On the futile quest for a precise delineation of BP's

Two related issues are examined in this section. The first is the suggestion that cognitive interest in the BP domain is limited because it is a microworld, akin to chess and tic-tac-toe: it does not scale up. The second issue concerns the attitude assumed by some researchers when confronted with this domain: "Let us define precisely what the domain is." In the following subsections, the first idea is shown to be incorrect, and the second one inappropriate.

2.3.1 Beyond Bongard's world

Bongard did not specify explicitly any restrictions on his problems, but some limitations can be inductively inferred. For example, all problems in his collection consist of black-and-white figures, suggesting black-and-white pixels if converted to digital form. His figures are also two-dimensional and, of course, static (lacking motion). A subtler characteristic is that the rules that solve BP's should be based on the *geometry* of the input only. This means that a geometer, using no culture-specific knowledge, should be in a position to understand the rule. Thus, a BP that shows triangles and rectangles is a valid one, but a "BP" that contrasts furniture with kitchen utensils is invalid. This principle follows inductively by examining all 100 problems designed by Bongard, although there are occasional instances in which it could be argued that he ventured slightly outside this "geometry only" principle (for some examples, see BP #69 in Figure 1.7, and BP #100 in Figure 2.11). Hofstadter's BP #108 (Figure 1.10) is another case in point. The examples that follow attempt to show that Bongard's world can be richer than first imagined, even adhering to the restrictions implicitly present in the domain.

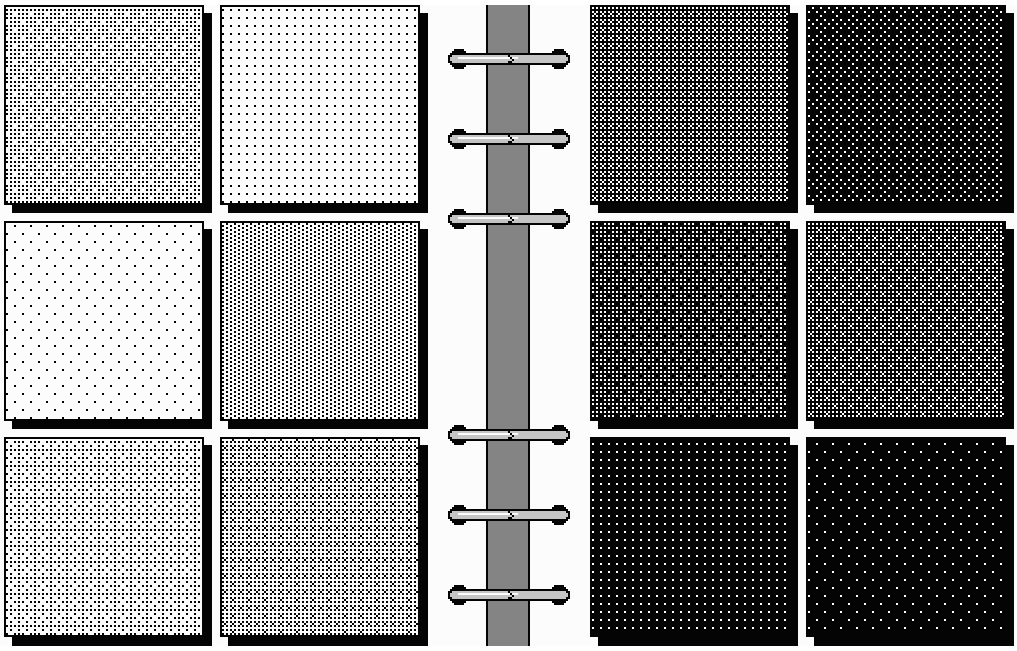


Figure 2.12: BP #196, with "shades of gray" ("colors"?)

BP #196 (Figure 2.12) is the first in a series of problems designed to exemplify the flexibility in the conception of a BP afforded by the domain. In BP #196 the textured area of each box can be seen as a shade of gray, if perceived as an average number of black pixels per unit of area (which is the metric that can be used in the solution of this problem). From a more abstract perspective, however, such textured areas can be construed as substitutes for colors. Naturally, adding *real* colors would lead to a further extension of the domain.

In chapter 10 it will be explained that Phaeaco can perceive not only shades of gray, but true colors of photographic quality (§10.1), and can represent them internally, although it cannot perceive complex real-life objects at present.

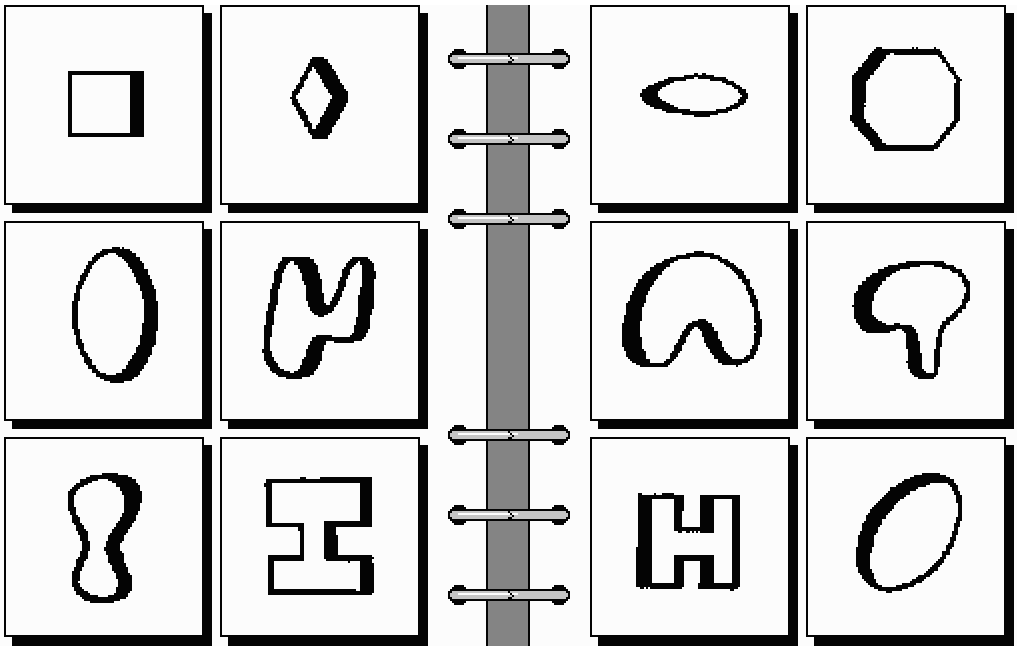


Figure 2.13: Objects with three dimensions in BP #63

Bongard experimented with the idea of three dimensions, but only within the overarching principle that the solution must be easily derivable from the two-dimensional geometry of the input. For example, in BP #63 (Figure 2.13), the geometric solution is that, on the left, the rightward sides of the objects are thicker, while on the right the leftward sides are thicker. Although the dichotomy "thicker vs. thinner" is sufficient for a correct description of the solution, the human visual system perceives immediately the thickness of each object in the third dimension, and interprets the thicker sides as shadows produced by the different orientation of the source of light on each side of the problem. Indeed, the concept of "shading" was Bongard's preference for expressing the solution ("shading thicker on the right vs. shading thicker on the left").

A problem that utilizes directly the ability of our visual system to perceive in three dimensions is the following.

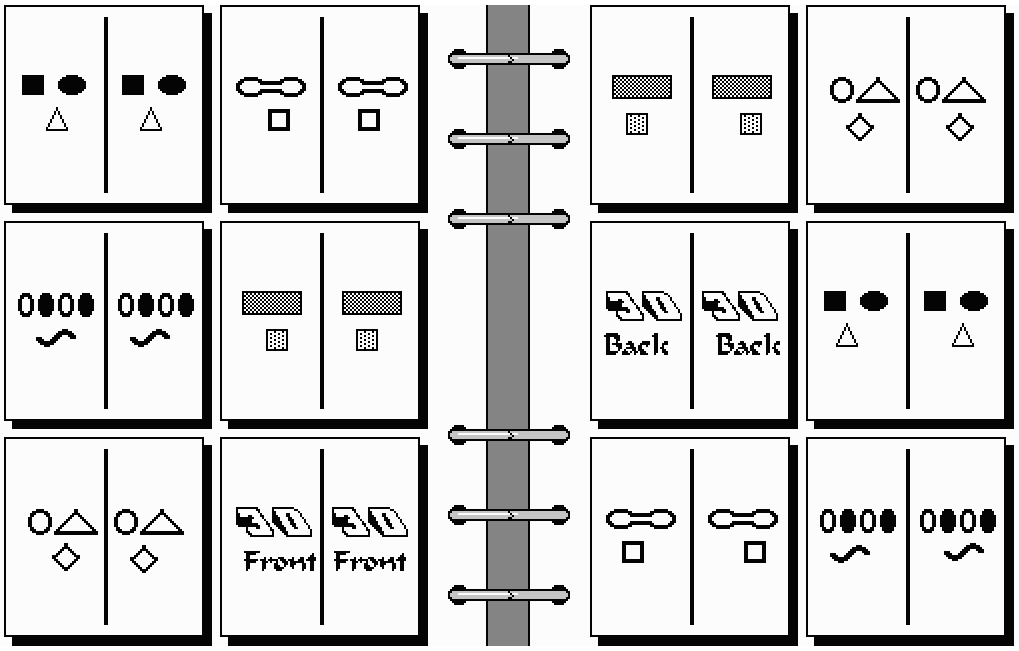


Figure 2.14: Direct perception of third dimension in BP #195

In BP #195 (Figure 2.14), each box contains objects separated with a vertical bar in a left and right group. Horizontally, the corresponding elements of each group are identical. In the left-side boxes of BP #195, the lower two objects are closer together than the upper two ones (the opposite relation holds in the right-side boxes). If the left group of objects in a single box is directed to our left eye, and the right group in the same box to our right eye, our visual system perceives a single group (with both eyes) within the box, where the lower object stands out in the third dimension, in front of the upper object. (On the right side, the lower object appears behind the upper one.) In neurological terms, our visual cortex records the *visual disparity* of the images of the upper and lower objects, and the magnitude of this disparity gives us a measure of the distance in the third dimension between the two objects (Thompson, 1993).

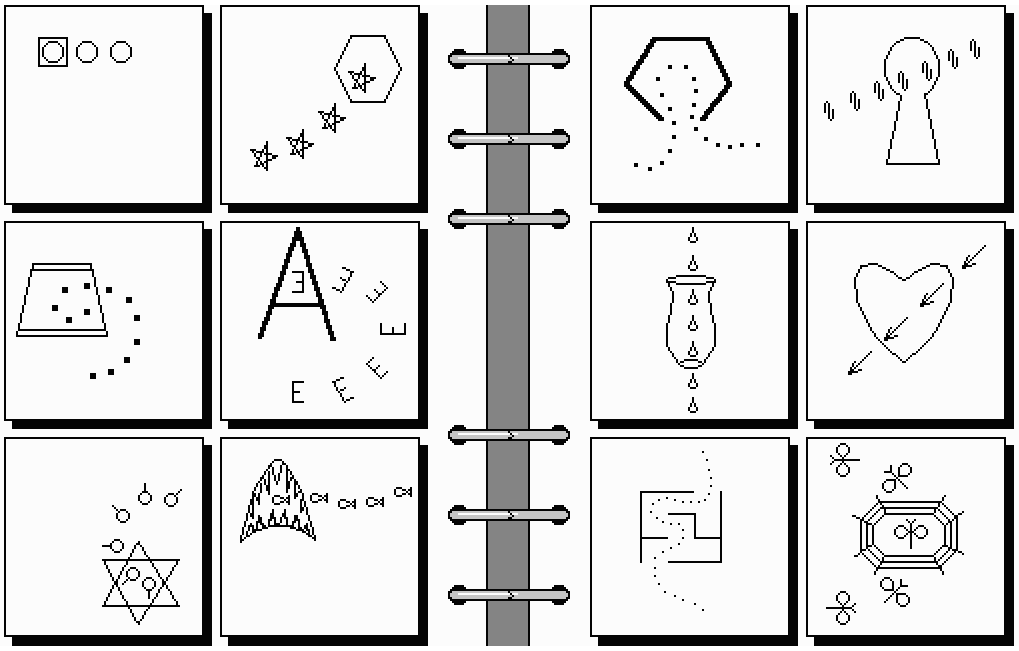


Figure 2.15: Motion in BP #198

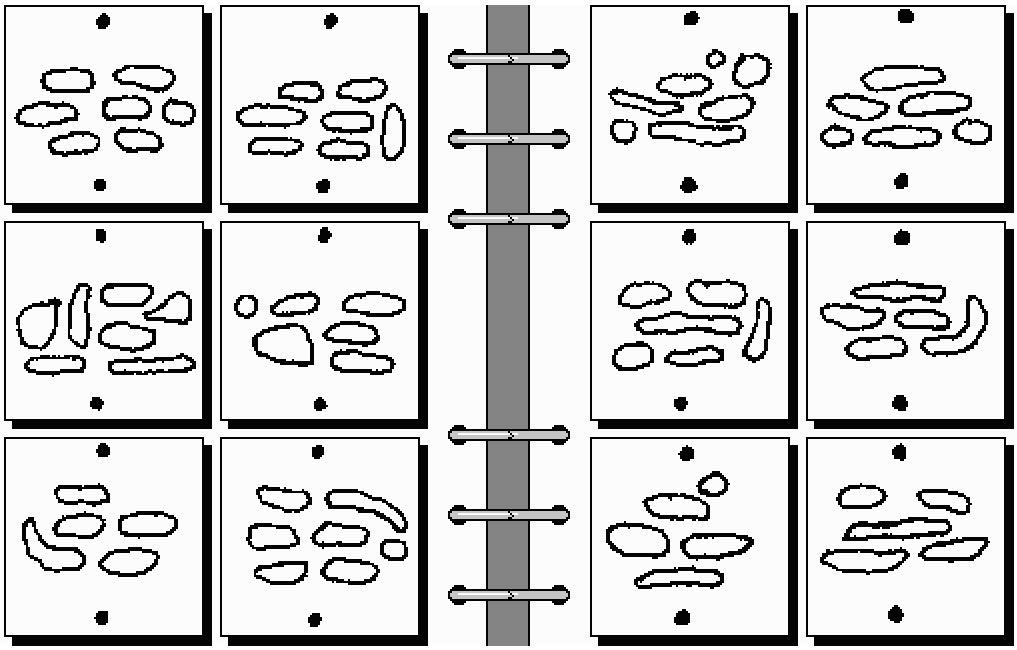


Figure 2.16: Navigation in BP #176

One might argue that if colors and a third dimension do not impose limits on what is explorable in the static world of BP's, motion does. But BP #198 (Figure 2.15) offers a counterexample in which motion is perceived as a series of snapshots along the trajectory of a moving object. The solution of BP #198 is that the "moving" object arrives and stays in a certain region on the left (or starts off from the region and moves out of it), but passes through the region on the right.

BP #238 (Figure 2.16) is related to motion, but of a different nature. Here we have a problem of *navigation*. In each box the lower dot is reachable from the upper dot with a (possibly curved) line that does not cross or touch any of the intervening "obstacles". On the left side, all dot-connecting pathways of minimum length are much shorter than the corresponding minimum-length pathways on the right side.

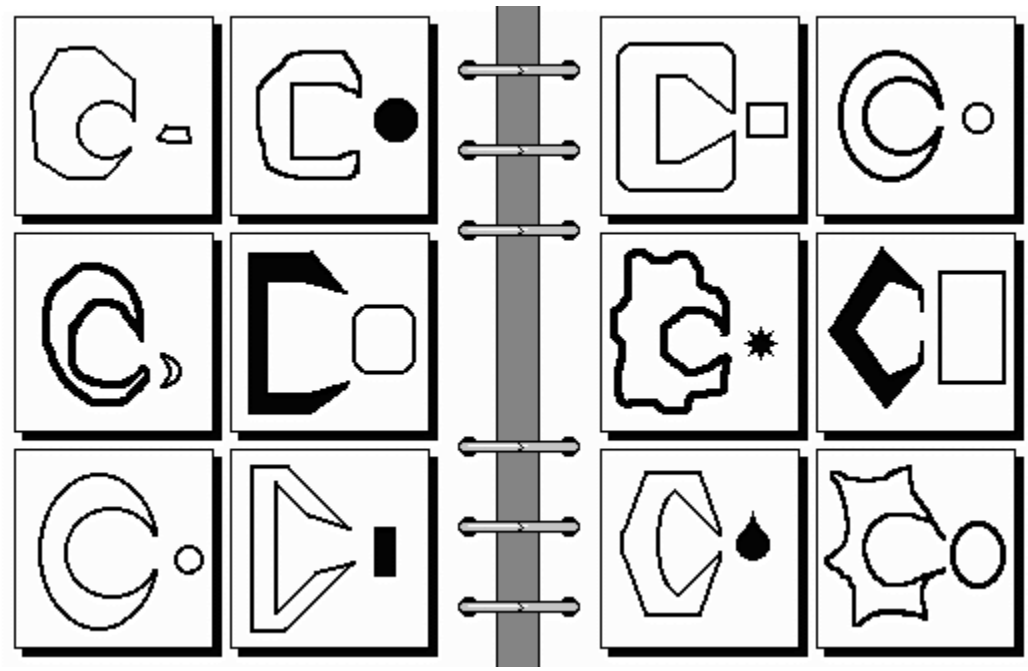


Figure 2.17: A different navigational problem in BP #175

The issue of navigation is encountered also in BP #175 (Figure 2.17), as trying to pass a smaller object through the "orifice" of a larger one. The problem here is not merely one of measuring the overall size of the small object and testing whether it is smaller than the "empty area" inside the larger object, because several boxes on the right side conform to this description. Nor is it one of simply gliding the smaller object leftwards: the objects of the boxes I-C and I-F must be rotated to pass through the orifice. To solve this problem a mental model of the motion of the smaller object must be made. We often encounter such puzzles in real life, e.g., having to move a car out of a parked position in a non-obvious way, or having to pass a large piece of furniture through the door of a room.

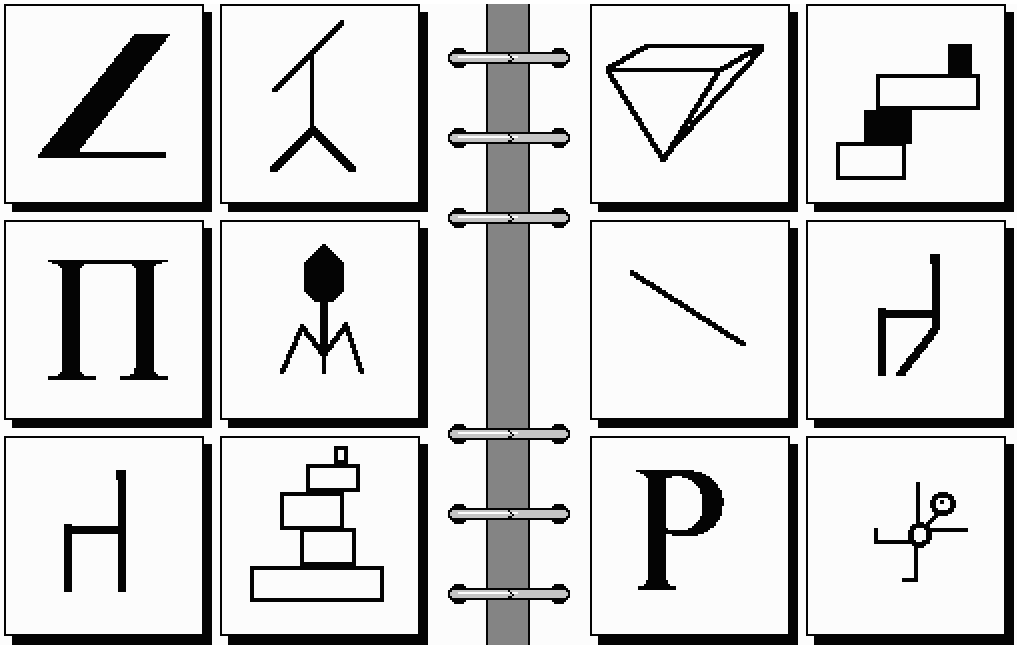


Figure 2.18: Motion combined with gravity in BP #199

Before leaving the subject of motion, consider BP #199 (Figure 2.18). Here the solution is that, if "gravity" were applied, the objects on the left would stay

put, while those on the right would topple over. Even though this appears to be a problem of physics, it is still solvable within the domain of geometry: each object (or collection of objects) has a center of gravity, the "barycenter"; if a vertical line through the barycenter intersects the "base" of the object, the object will remain stable; if not, it will topple.

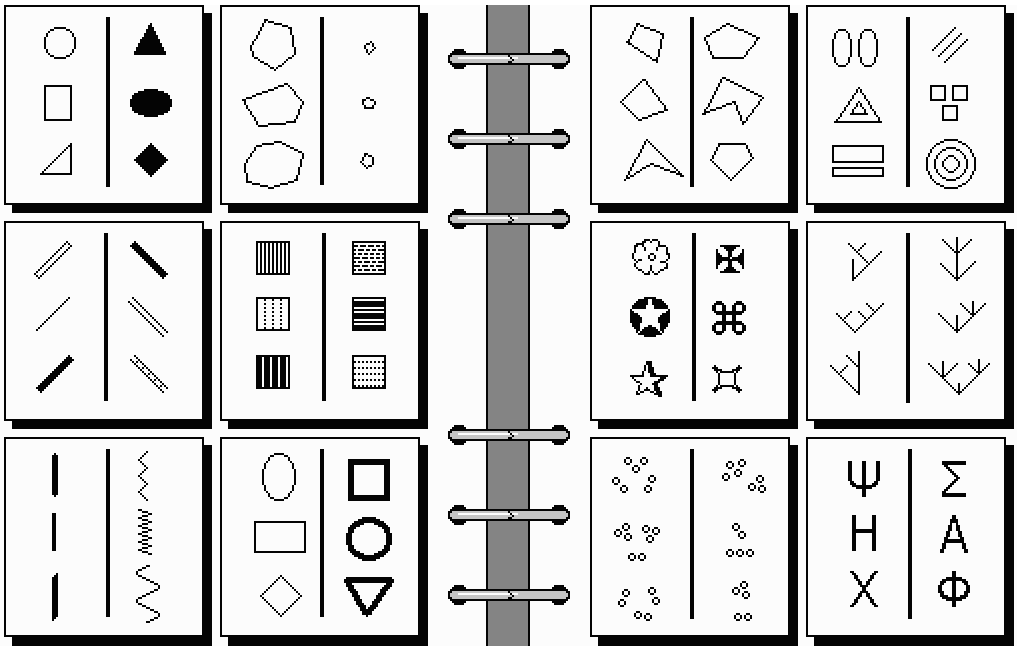


Figure 2.19: BP #200 is a Bongard problem about Bongard problems

Finally, if we give up geometry as a domain imperative, even the abstract idea of *self-reference* is expressible. For example, BP #200 (Figure 2.19) is a problem in which each box itself is a "Bongard problem". The solution of BP #200 is as follows: on the left the solution of the Bongard problem within each box is based on the value of a simple *feature* (texture, area, slope, etc.); while on the right the solution of the Bongard problem within each box is based on *numerosity* (number

of sides, objects, branches, etc.). BP #200 could thus be termed a *meta*-Bongard problem.[6]

Thus, the domain of BP's can be utilized to experiment with a variety of cognitive dimensions that do not immediately appear amenable to such experimentation. Although, to be sure, there are many issues in cognition that would seem to be difficult to address via the BP domain (e.g., the robotic experience of interacting with a physical world, speech recognition and production, music perception, and many more), the hope is that the domain of BP's will prove to be suitable for addressing certain *core* issues in cognition, on which most other issues are based.

2.3.2 Defining the domain

Some researchers, when confronted with the BP domain, try to define precisely (mathematically, if possible) the problem at hand.[7] Usually this is the case with people who, having strong mathematical training and background, are accustomed to giving precise definitions to the objects they study. Under this approach, anything undefined (or "ill-defined") cannot be an object of scientific inquiry. Specifically, they would like to describe a function $F(x)$, in which x is a piece of visual input (a $m \times n$ matrix of pixels, for example, where each pixel is a number from a suitable range of possible numbers), and F returns true if x is a "valid" BP, and false otherwise. In other words, F would characterize some inputs as BP's, and all others as non-BP's.[8]

---

[6] The author and Joseph A. L. Insana have created an entire collection of meta-BP's (and even meta-meta-BP's) (Foundalis, 2001).

[7] Thanks to Ralf Juengling (personal communication) for bringing up this issue.

[8] Hence, F is a *total recursive function* in computability terms, or a *decidable algorithm*.

The problem with this attitude is that it creates an arbitrary group of objects, christened "valid BP's", or "the BP's we want to study", whereas no such group exists objectively in reality. What exists in reality is the human mind and its fascination with some puzzles. The precise delineation of those inputs considered to be valid BP's fundamentally changes the nature of the cognitive quest. Instead of: "Here is a set of problems that appear interesting to people; let us explore the cognitive mechanisms people use to solve them", the quest becomes: "Here is an *ad hoc* set of problems; let us write a program that can solve all of them". In this way, rather than aiming at cognitive exploration, the solution of BP's becomes a self-serving end: the researcher sets out to write a program so as to claim that the solution of BP's in "the domain" (the arbitrarily decided one) has been fully automated — and then possibly to move on to different domains. The pitfalls of this approach are examined in more detail in chapter 4.

CHAPTER THREE

Universality and Objectivity of BP's

3.1 Are BP's cross-cultural?

The present thesis is a product of a particular, but dominant, human culture, the so-called "Western" one. The scientific world-view of this culture can trace its roots back to the early explorations of nature by Egyptian and Mesopotamian cultures that emerged at least 7,000 years ago. The particular geometric concepts that are assumed, in the present text, to be fundamental and well-understood by everyone (e.g., "triangle") were explored to a great extent by the Greeks, as early as 2,600 years ago. Subsequently geometry became part of primary education, and was considered a foundational element of the arts (painting, sculpture, architecture) and sciences (physics, astronomy). Cognitive science, however, seeks to explore the nature of cognitive universals[9], not of concepts assumed as "givens" by a particular culture. Is the concept "triangle" understood in a universal way across cultures, or is this impression a bias of our "Western" culture? The following subsection attempts to shed some light on this issue.

[9] In principle, cognitive science seeks to explore cognition *in general,* not necessarily the only known implemented example of it, which is constrained by biological evolution. But without knowing the answer to the question of whether cognition is implementable in a different medium or not, it is best to confine the scope of questions asked about "universals" in this text to the single known instance, i.e., human cognition in the context of human culture.

3.1.1 Geometry as perceived by peasants and indigenous people

In 1931–32, the Russian psychologist Aleksandr R. Luria and his team conducted a series of studies probing the cultural foundations of cognitive development — that is, the extent to which culture (and language) influences the formation of concepts. His subjects included illiterate peasants from "the remoter regions of Uzbekistan and Kirghizia, in the *kishlaks* (villages) and *dzhailaus* (mountain pasturelands) of the country" (Luria, 1976). Among the issues examined by Luria was perception of basic geometric forms. The views of his subjects are interesting in the context of the discussion on human geometric universals.

In one of the experiments (conducted in far-from-ideal conditions), Luria showed to his subjects a list of drawings similar to the following.

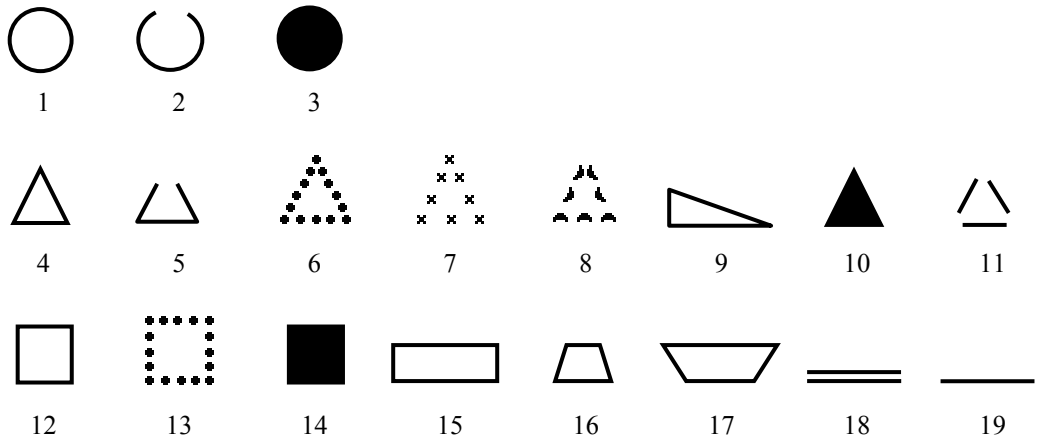


Figure 3.1: Geometrical figures presented to Luria's subjects

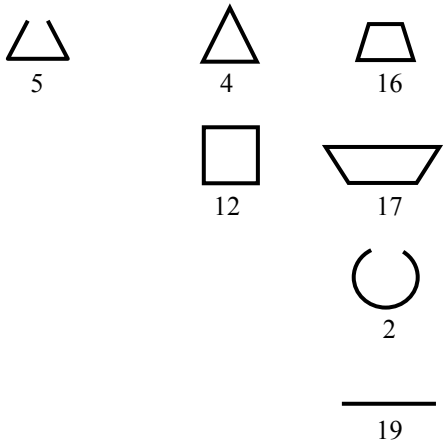
He then asked subjects to name each figure. The subjects were coming from four different population groups: ichkari women (illiterate), women students in short preschool courses (barely literate), collective-farm activists, and women students at a teachers' school. Only the last group of people — the most educated ones among all subjects — used mostly geometrical names to name the figures

either directly ("circle", "triangle", "square", etc.), or through descriptive phrases ("something like a triangle", "a square made of dots"). Ichkari women — the least educated group — never referred to the geometrical shape of the figure, preferring object-names instead. Thus, they would call a circle a moon, a watch, a bucket, and so on; a triangle would be a *tumar* (an Uzbek amulet); and a square would be a house, a door, a mirror, or an apricot-drying board. Table 3.1 summarizes the naming preferences of the various groups of subjects.

Subject group	*Number of subjects*	*Geometrical names*	*Object-like names*
Ichkari women	18	0.0 %	100.0 %
Women in preschool courses	35	14.7 %	85.3 %
Collective-farm activists	24	41.0 %	59.0 %
Women in teachers' school	12	84.8 %	15.2 %

Table 3.1: Naming preferences among Luria's subjects

Especially interesting were the answers obtained from illiterate subjects when asked to group the figures together. For example, an illiterate woman, age 24, from a remote village, formed the following groups.



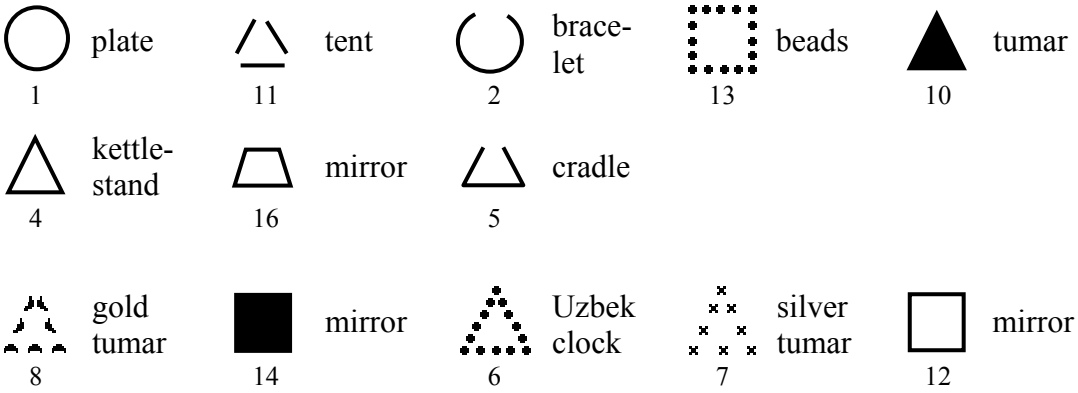
"These are all tumars."

"That's a glass and that's a glass, but with a wide bottom."

"This moon should go by itself."

"This thread should go by itself."

Another subject, age 19, an ichkari woman, gave the following names.



When asked to group the figures, she put the "valuable tumars" together (7 and 8), and also the "mirrors" (12, 14, and 16), declaring that none of the other figures were similar.

Answers along the same lines were obtained by other illiterate subjects. On the contrary, subjects who had even minimal exposure to formal education did make occasional use of geometric names, and the percentage of such use increased with the sophistication of the educational background (Table 3.1).

However, the experimental method used by Luria and his colleagues, particularly their reliance on conversation and linguistic descriptions, is highly suspect. Recall that Luria performed his experiments in the ex-Soviet Union at a time when it was considered imperative to amass evidence for the necessity of educating the nation's largely peasant population and working class. Could further, language-independent experiments paint a different picture?

Indeed, this idea seems to be confirmed in recent experiments in which a team of researchers led by Stanislas Dehaene studied the indigenous Amazonian tribe of Mundurukú (Dehaene, Izard *et al.*, 2006). Children and adults of the tribe, who had received no schooling and had no experience with graphic symbols, maps, or a rich language of geometrical terms, were asked to find which figure from a

group of six figures did not belong to the group. For example, the upper right box in Figure 3.2 shows a non-trapezoid, whereas all the other boxes contain trapezoids. The participants were asked to point to the "weird" or "ugly" figure.

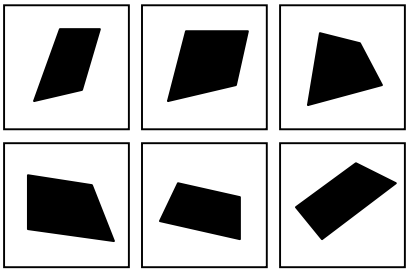


Figure 3.2: Sample problem of type "find the odd-man-out" used by Dehaene *et al*

According to the study, Mundurukú adults, children aged 6, and also a control group of 6-year-old American children all performed similarly. Only American adults performed significantly better. However, all groups showed a shared competence and understanding of basic geometrical concepts.

3.2 Objectivity of the difficulty of various BP's

It is almost certain that performance in Bongard's domain correlates strongly with prior education. But given a group of people from a culture who are educated enough to appreciate the domain a question that arises is whether there is any property of the performance of BP-solving that can be measured objectively.

With this in mind, an experiment was administered to American college students, aiming at establishing a more or less evident result: that some BP's are easy, some others are hard, and that all intermediate degrees of difficulty are possible. Each particular BP-solver usually acquires a personal sense regarding the difficulty of each problem they attempt to solve, but what is required is some

objective measure of this difficulty. A secondary aim of the experiment was to be in a position to compare Phaeaco's performance with that of human solvers (although see chapter 4 for an important caveat regarding the conclusions drawn from such a comparison).

3.2.1 An experiment with American college students

Subjects

The subjects were 31 students of Indiana University, at different levels in a four-year bachelor's program. Their educational background varied, but none was studying for a degree in mathematics, physics, astronomy, computer science, or chemistry, or any curriculum related to those disciplines. They had been exposed to high school geometry at various levels of rigor. Their age ranged between 19 and 25 years.

Method

Subjects were presented with a special "Experimenter" session of Phaeaco, in which the program presents BP's in a predetermined order. The 12 boxes of the problem are initially covered with a mask, and when the subject signifies they are ready (by pressing the spacebar) the mask disappears and the BP is shown, while a chronometer starts recording the time. As soon as the subject thinks they know the solution of the problem they hit the spacebar, and the mask hides the 12 boxes again, while the chronometer stops and a dialog-window pops up, prompting the subject to type (in natural language) the rule for the left and right side. If the subject has no answer, they may signify this by clicking on a "Give up" button. The subject can also "Take another look" (another button on this window) at the problem, in which case the chronometer resumes counting from where it stopped earlier. This feature, which can be repeated any number of times, was deemed

necessary because subjects are instructed to hit the spacebar as soon as an idea for a solution occurs to them, without attempting to express the idea linguistically first. Sometimes the subject hits the spacebar too soon, and wishes to reconfirm the idea before writing it down on the dialog-window, hence the need to take another look. As soon as the subject moves on to the next problem, the overall time spent thinking for a solution on the previous problem is recorded in a file. The subject cannot return to work again on a not-answered (skipped) BP once the next BP is shown. Each session lasts a full hour, and contains the first 100 BP's, (see Figure 3.5). Subjects try to solve as many BP's as they can within this time.

Before starting the actual session, the subject goes through a practice run that includes a small number of BP's (up to five), and is designed to familiarize the subject with BP's and the interface. These BP's make use of a few elementary concepts (such as "outlined", "filled", "large", "small", etc.). The following instructions are read to the subject when the first such BP is shown:

> "What you see here is a 'problem' that gives you six boxes on the left, and six boxes on the right. Each box contains a figure, or figures, and your task is to find out why those figures on the left have been separated from those on the right. There is some rule, some underlying principle, by which the figures have been separated like that, in two groups. For example, it's rather evident what the rule in this example is, right? [Figure 3.3] Those figures on the left have more white than black, and those on the right have more black than white. Correct? [all subjects invariably nodded at this point.] As soon as you discover a rule that describes the contents of the boxes on the *left* side, and *none* of the boxes on the right side, you hit the spacebar. Please do so, now. [The subject

hits spacebar, the mask hides the problem, and the dialog-window pops up.] On this window, you can type your answer. For instance, here you can click on the box for the left-side rule, and type simply "white", or "outlined". On the right-side box you can type "black", or "filled". Sometimes the rule for the right side will be simply the opposite of the left; in that case you can leave the right-side box blank. Use simple English, and type only what is necessary to let me know that you understood the rule — no full syntax is needed."

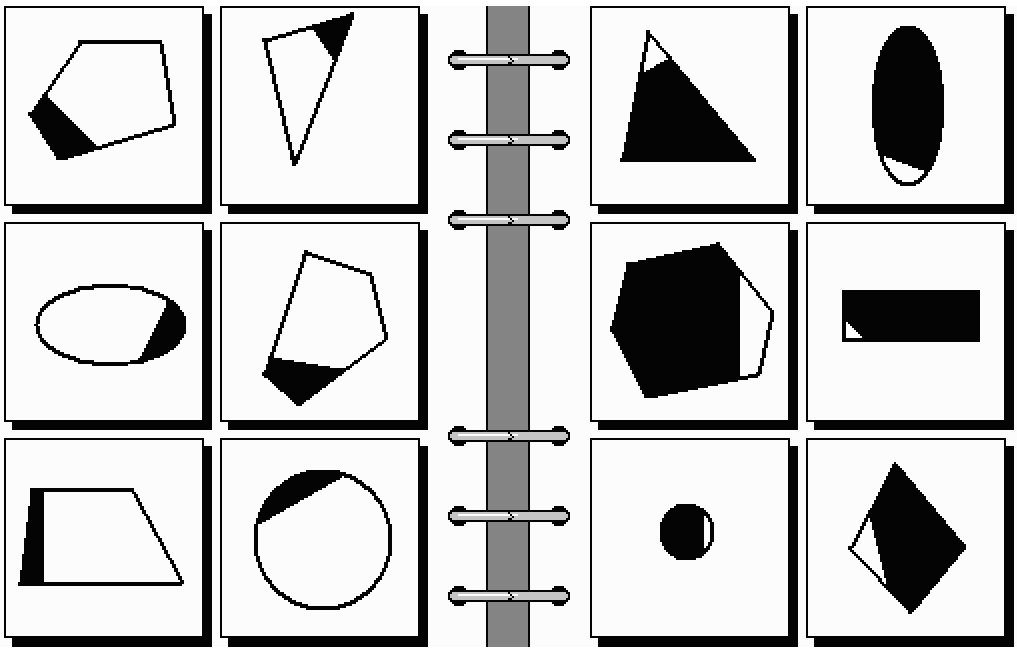


Figure 3.3: The first BP in the familiarization session

After explaining the "Take another look" button, and moving on to the next problem, the subject was cautioned on the following issue.

"Notice that, in the rule you come up with, you can't make use of the position of the box within the matrix of six boxes. For example,

> you cannot say, 'The figure at row one, column two, is such-and-such', or, 'All figures of the first column are…', and so on. Your rule must treat all six boxes as a *lump,* with no particular order."

After making sure the subject understood the above, one more subtle point was brought to their attention.

> "Now, here is another thing you cannot do with your rule. Take a look at this problem: [Figure 3.4] You can't say here, for example, that there is no black on the left side, but there is some black on the right. Your rule must be true *for each individual box* on the left, and false for each individual box on the right. If you say, 'white only' for this problem, your rule will be wrong, because there are several boxes on the right with white figures only."

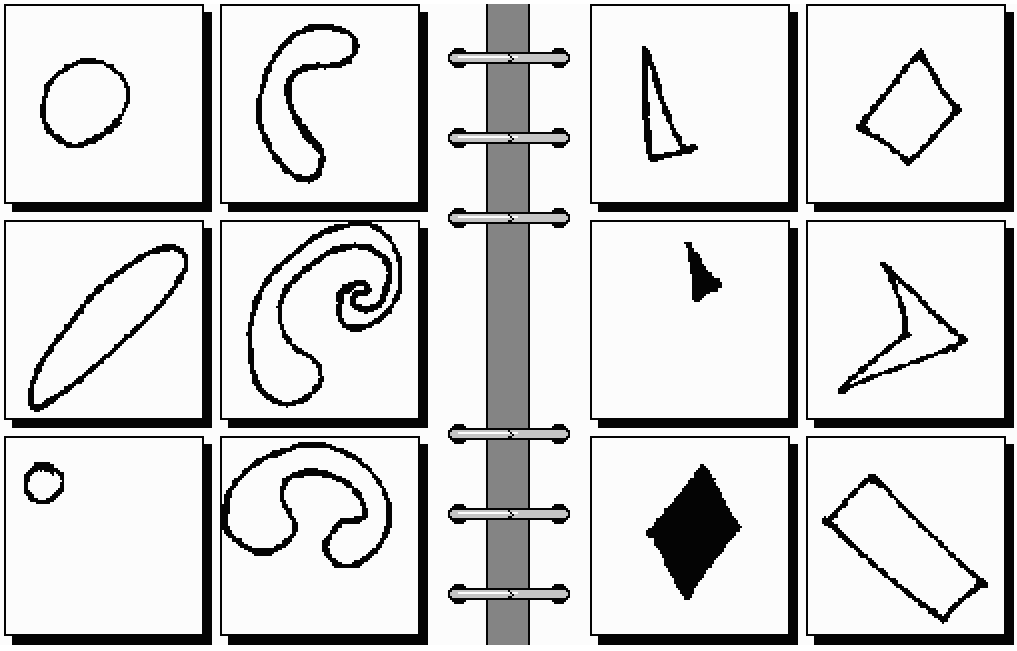


Figure 3.4: A familiarization-BP for cautioning the subject on the nature of rules

Although all subjects agreed that they understood this last point, in the process it was found that a few of them did not heed it. Their answers to such problems were removed from the answer set. (Their answers to other BP's that are immune to this pitfall — and those are the majority — were retained, because in such cases the pitfall seems to be irrelevant.)

Once the subject agreed they understood the instructions, the following — very significant for the purposes of the experiment — remark was announced.

> "You'll receive a reward of 10¢ for each problem that you solve correctly. Later I will find the number of correct answers and write a check to you for the total amount. [It was necessary to explain this because in an earlier pilot study some subjects thought the reward would be fake.] Now, *your task actually is to maximize your profit*. This means it is up to you to decide for how long you'll be thinking on each problem. Because if you think for too long, you might end up with no time to do some easier problems that lie ahead; while if you give up too soon, you'll miss this problem, which you might be able to solve if you could think for a little longer."

The additional element of monetary compensation was deemed necessary for the following reason. These subjects, being enrolled in a course in psychology, were all *obliged* by the Department of Psychology to participate in one experiment of their choice during the semester. Thus, since they did not volunteer their participation, and because the nature of this experiment required them to think intensively for one hour, there was the danger that several of them would simply click on "give up", problem after problem, in order to satisfy the said

requirement. The monetary reward provided the necessary motivation for an honest performance.

Finally, the exact sequence of BP's used in the real session is given below. The order of problems differs from Bongard's original one (1–100) for two reasons. First, as mentioned in §1.2.5, Bongard often used the idea of *priming* a concept, by first introducing it in a problem and then using some variation of it in subsequent problems. The experimental list tried to minimize priming and concept interference, allowing relatively "pure" results to be obtained. Second, a pilot study showed that subjects generally required a small number of problems to "warm up" in the real session, *in addition* to the BP's of the familiarization session. For this reason, the first five problems of the real session were present only for warming up, and their timings were discarded by the experimenter.

264	160	166	167	173	2	6	1	23	3	5	4	9	8	10
7	11	15	21	39	56	12	24	22	36	85	33	34	38	40
48	47	45	77	51	61	71	81	84	96	89	83	86	91	97
87	88	95	82	78	75	72	70	65	67	66	62	57	50	49
53	46	41	37	35	31	32	25	30	29	13	14	16	17	18
19	20	26	42	27	43	28	44	52	54	93	55	58	63	59
64	60	68	69	73	74	76	94	79	80	90	92	98	99	100

Figure 3.5: Exact sequence of BP's tested (read by rows; first "real" BP is #2, then #6, etc.)

Results

Appendix A, which lists 200 BP's, also gives the results of this experiment for each problem, showing the numbers, mean times, and standard deviations for correct answers, as well as the numbers and mean times of no answers and wrong answers. Most subjects (17) used the entire hour and answered fewer than the 105 BP's listed in Figure 3.5, but nearly half of the subjects (14) were fast enough to complete the task in less than an hour.

	Correct answers			No answer		Wrong answer	
	time (sec.)	95% conf.	num.	time (sec.)	num.	time (sec.)	num.
BP #100	5	±2	14				
BP #1	7	±2	31				
BP #15	8	±1	27	30	3	7	1
BP #95	8	±2	30	17	1		
BP #3	8	±2	28				
BP #94	8	±2	15				
BP #34	9	±2	30			19	1
BP #23	9	±2	30			20	1
BP #97	9	±2	29	18	2		
BP #63	10	2	15	19	1		
BP #9	10	3	31				
BP #25	10	3	22	5	4		

Table 3.2: Easiest BP's

In agreement with intuition, it was found that the easiest BP's are as shown in Table 3.2. Somewhat surprising was the position of BP #94 (Figure 3.6).

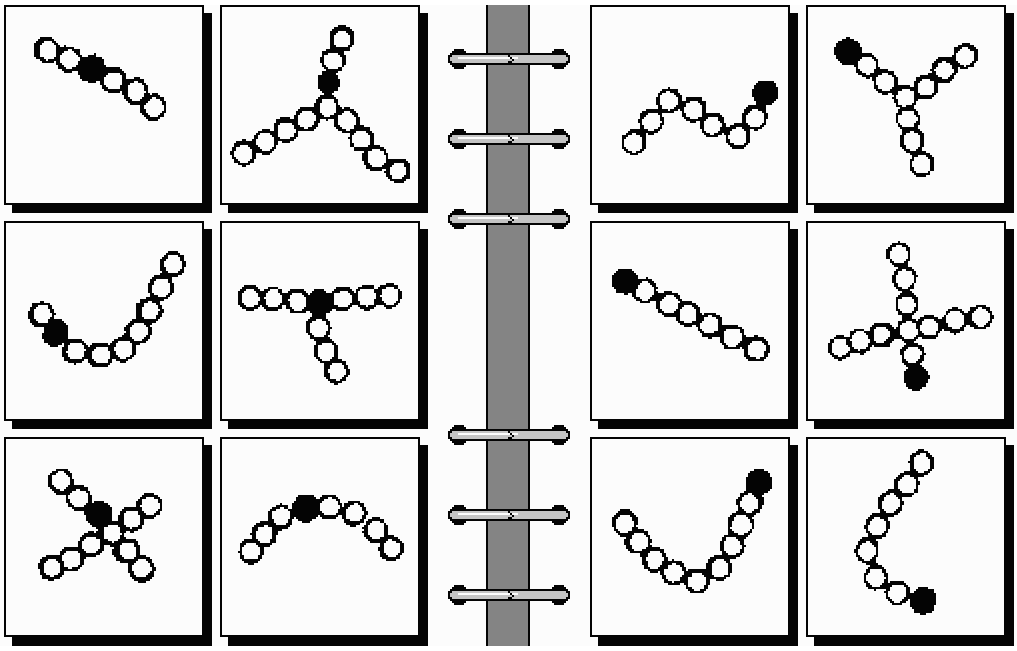


Figure 3.6: BP #94, a surprisingly fast-solved problem

Also surprising was the slow response time for BP #2 (Figure 1.3; "large vs. small"): 28 subjects answered correctly with an average time of 14 sec. and a 95% confidence interval of ±4 sec. This can probably be explained by the early appearance of BP #2 in the experimental list, very close to the "warm up" zone. Overall, however, there were few other surprises in the collected statistics of the problems. The fastest-solved problem, BP #100 (Figure 2.11), shows the effect of memory on pattern recognition (all subjects interpreted the Cyrillic letter "Б" on the right side of this problem to be the English letter "b"). Comments on the statistics of other problems will appear throughout the text, as their cases are encountered.

3.3 Summary

In conclusion, the experimental observations discussed in this chapter suggest that people, when properly questioned, show an innate, cross-cultural understanding of basic geometric concepts. Understanding precisely the task of solving BP's can be mildly confusing even for American undergraduate students. Nonetheless, the collected statistics, particularly the rather small variance in response times, imply that there must be some fundamental cognitive mechanisms at work among those people who do understand the task and manage to solve at least some of the problems. This, in turn, suggests that the idea of exploring those mechanisms by creating a program that automates the task of BP-solving is well justified. Automation is the subject of the next chapter.

CHAPTER FOUR

Automation of BP-Solving

4.1 RF4 and the problem of input representation

Few efforts have been made to explore computationally the BP domain. One attempt that deserves special mention was outlined in a publication that described the merits of a general-purpose search algorithm in AI, called RF4 (Saito and Nakano, 1993). The algorithm searches a space of formulas of first-order logic by performing "a depth-first search on the basis of five criteria for pruning undesirable formulae and five transformation rules for combining formulae" (*ibid.,* from the abstract). Saito and Nakano examined a variety of domains in which RF4 reportedly excels, and one of these domains was that of Bongard problems. Saito and Nakano claimed that RF4, which was implemented in a personal computer in the C programming language, "solved 41 out of 100 Bongard problems within a few seconds for each problem". This bold statement implicitly informs the reader that the BP domain is merely a collection of puzzles and moreover that it has been successfully tackled by RF4. If 41 BP's could be solved within a few seconds, then perhaps an hour or two would suffice for all of the others. Thus, the domain of BP's would have been automated by 1993!

The article also provided information regarding the method by which BP's are encoded to be given to RF4. Each figure in a box of a BP is represented by a first-order logic formula. So, for example, consider Figure 4.1.

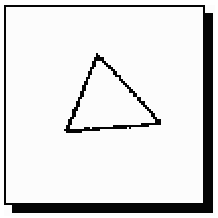


Figure 4.1: A single triangle, out of context

The triangle in Figure 4.1 could be represented by a formula similar to the following:

$$\textit{polygon}\ (X) \wedge \textit{outlined}\ (X) \wedge \textit{angles}\ (X) = 3$$

Figure 4.2: Possible representation of a triangle in RF4

It is not known whether the above formula would be the exact representation of the triangle in Figure 4.1 for RF4, because Saito and Nakano do not provide examples of input representation from the domain of BP's. It is a reasonable guess, however, because Saito and Nakano provide as an example the *solution* to BP #6 (Figure 1.1), which they write as follows:

$$\textbf{forall}\ B\ \textbf{in}\ \textit{boxes,}\ \textbf{forall}\ X\ \textbf{in}\ B,\ \textit{angles}\ (X) = 3 \rightarrow \textit{class1}$$

In other words, if a figure X of a box B has three angles, then it belongs to class 1 (left side of boxes of BP #6). The plausibility of the first-order formula in Figure 4.2 is inferred from the observation that if a triangle were represented by something as simple as *angles* (X) = 3, then there would be no way to distinguish between outlined and filled triangles (hence, *outlined* (X)), nor between closed and open shapes with three vertices (hence, *polygon* (X)).

Further elaboration of the previous argument for the plausibility of the formula in Figure 4.2, however, reveals the problematic nature of this approach

for input representation. Why should the location of the triangle within the box be omitted? After all, the solution of BP #8 (see Appendix A) refers explicitly to the location of a figure in relation to its containing box. Hence, the coordinates of a center of the triangle (e.g., its "barycenter") should be included in the formula:

$$\textit{barycenter}\ (\mathrm{X}) = \mathrm{p} \wedge \textit{polygon}\ (\mathrm{X}) \wedge \textit{outlined}\ (\mathrm{X}) \wedge \textit{angles}\ (\mathrm{X}) = 3$$

Otherwise the representation makes the *a priori* assumption that the center of the figure is irrelevant; but one cannot know what is relevant in the solution until one solves the problem.

For similar reasons, the list of terms that describe the triangle in Figure 4.1 must grow to include the width, length, and slope of each side, the coordinates of each of the three vertices, and possibly more. Indeed, even the fact that this is a *triangle* is not known before solving the problem, as can be seen in BP #85.

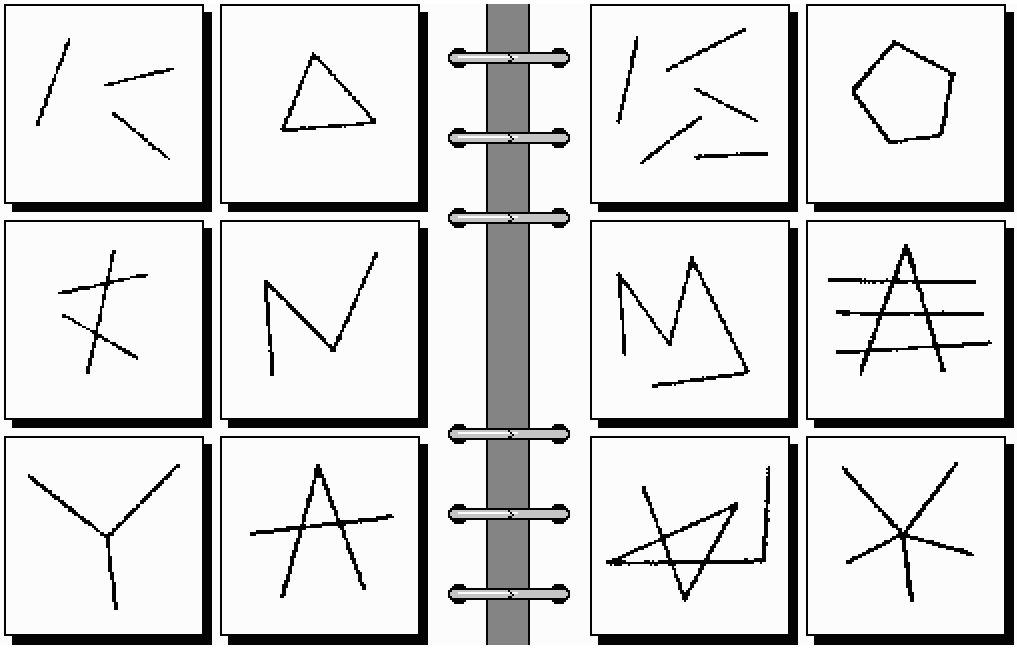


Figure 4.3: The triangle of Figure 4.1 in the context of BP #85

In BP #85 (Figure 4.3) the box of Figure 4.1 is included as box I-B. Yet the solution of this problem does not use the concept "triangle" at all; instead, it uses "three lines". On one hand, a logical formula such as the one in Figure 4.2 (or any of the extended ones proposed above), which omits the crucial term *lines* (B) = 3 (where B is a *box*), and instead describes the input as a triangle, makes a wrong representational assumption that would lead the system to fail to find a solution. On the other hand, including the term *lines* (B) = 3 would prematurely reveal the solution to the system.

One might argue that a good logical system with theorem-proving abilities should be in a position to infer that there are three lines in the figure, given that the rest of the formula describes a triangle. Though this is true for elementary geometric structures such as triangles, the BP domain is far too complicated to allow formal deduction of properties in general.

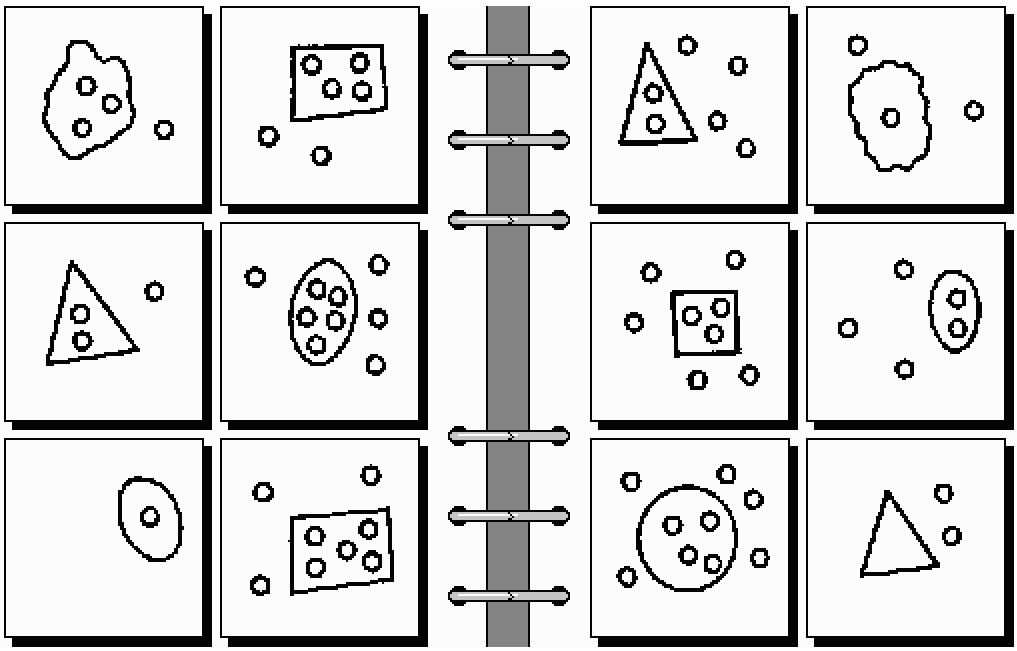


Figure 4.4: BP #29, where shape is irrelevant

Consider BP #29 (Figure 4.4), included in Saito and Nakano's article. The irregular shape that encloses three circles in box I-A is too complicated to be described accurately by logical formulas of reasonable length. Thus, a predicate such as *closed_region* (X) might appear reasonable enough to be included in the representation. This choice makes the assumption that the particular shape is irrelevant. This is correct in BP #29, because the solution involves counting the circles that are inside and outside the larger figure. How can one know, however, before solving the problem, that the shape of a figure is irrelevant? Other BP's use shape as an integral element of their solution.

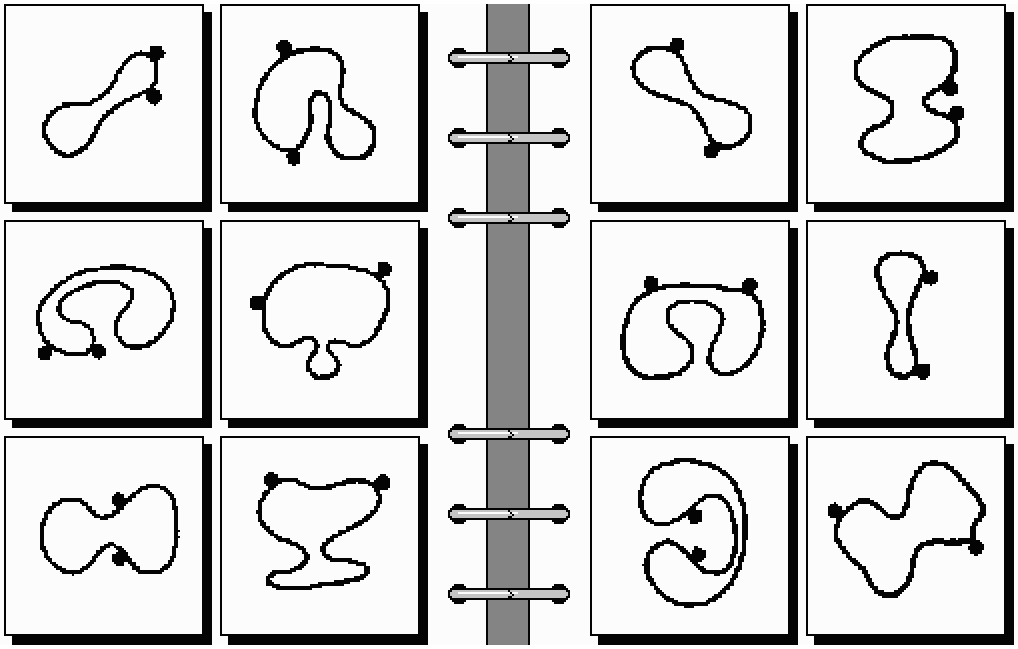


Figure 4.5: BP #20, where shape is important

In BP #20 (Figure 4.5), for example, a predicate such as *closed_region* (X) is useless: the solver must perceive the "neck" and the two bulges in each figure before noticing the placement of the two dots in relation to the two bulges. But it

is not possible to infer the existence of bulges in such figures from predicates such as *closed_region* unless an explicit description of the curve that forms the region is given — for example, in terms of a long sequence of predicates that describe short pieces of straight lines that meet each other end-to-end. Thus, we end up with two choices.

- Either the logical description is "honest", but too long and unwieldy for manipulation as a predicate calculus formula,
- or the logical description is short but "dishonest", giving away some predicate crucial for the solution. (*bulges* (X) = 2 would be such a predicate in the previous example.)

Overall, the conclusion is that logical formulas are the wrong medium for representing the input in BP's. Such formulas require a human "helper" who, knowing already what the solution is, uses just the right predicates for describing the input. If we assume that such a human helper is unavailable, then the most straightforward way to represent the input is to adopt Phaeaco's approach (and that of most other visual processing programs), which is to include explicitly all the pixels that form the image (the rectangular matrix of which can be the output of a camera, scanner, etc.), and let the program *discover* any features or relations by itself, unbiased by the preprocessing of human helpers. Otherwise we have not an automated system but a semi-automated one, in which human cognition plays an integral role.

An additional critique of RF4 from a slightly different perspective has also been offered by Alexandre Linhares (Linhares, 2000). In his paper, Linhares emphasizes the importance of the ability to re-parse the input under pressure (as in box I-B of BP #85, Figure 4.3), an ability missing entirely from RF4.

4.2 Maksimov's combinatorial approach

A different approach to solving BP's from that of RF4, at least in some respects, was that of V. V. Maksimov, a student of Bongard's (Maksimov, 1975). In the early 1970's, Maksimov used bit-mapped, black-and-white images as input to his program, and what is most admirable is that he did so in spite of the severe technical restrictions that were present in his computational system. The total size of his computer's memory was a mere 4000 64-bit words, occupied mostly by his program, and an additional 4000 words for data manipulation. Each input box had a resolution of 45×64 pixels. In total, his computer had 64K of memory in modern terms — a volume available in home computers in the West by the mid-80's.

Even with these meager computing resources, Maksimov managed to design a system that, according to his report, matched the performance of human subjects, at least to some degree. It is instructive that he achieved this even though, as we shall see, the working principles in his system were far from cognitively plausible.

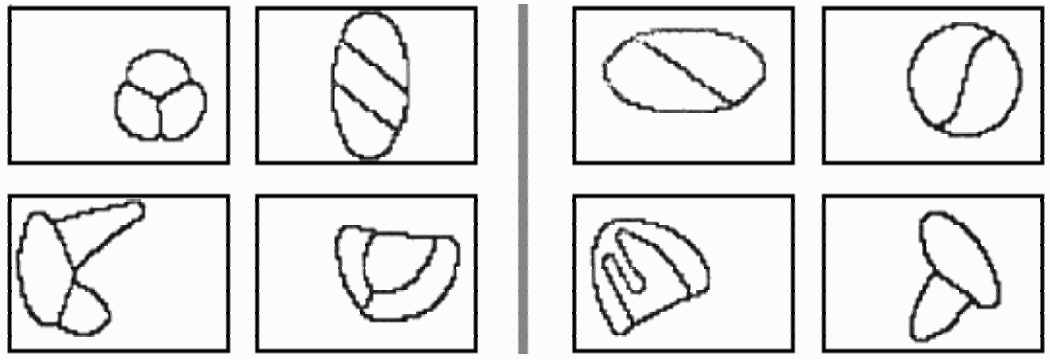


Figure 4.6: MP #13, three vs. two closed regions (four boxes per side)

A disclaimer must be made before proceeding: Maksimov's system did *not* solve any of the 100 BP's in Bongard's collection, but a set of 48 specially-designed problems. They will be called *Maksimov problems* (MP's) in this

section, to differentiate them from more "regular" BP's. Each MP could have any number of boxes on the left and right side; the usual number was six, but MP's with four or eight boxes per side were common (Figure 4.6). Sometimes, in order to gauge the performance of the system, an MP was simply a repetition of the previous MP, but with an additional box per side, retaining the solution. At other times a quite different type of problem was included, in which a number of boxes of unknown category were to be classified into one of two categories, which were hinted at by only two sample boxes, given at the upper-left and upper-right corner of the input (Figure 4.7). Maksimov called such problems "training sets", and their purpose was to incrementally teach the program the two categories by letting it receive feedback from a tutor.

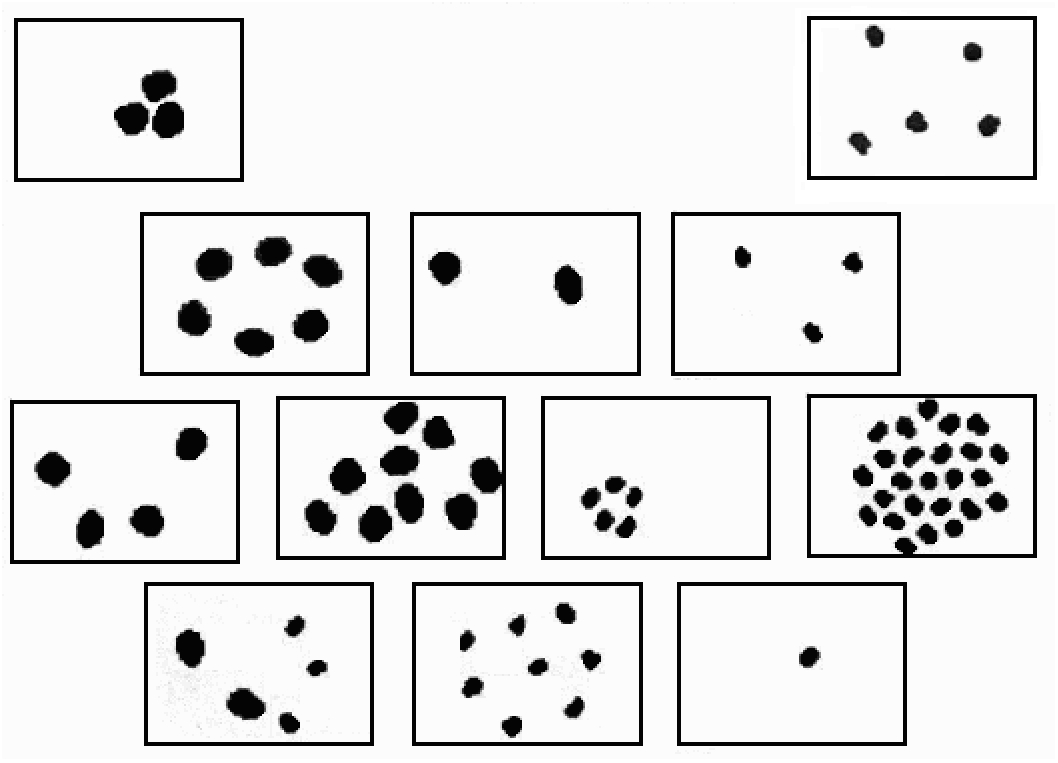


Figure 4.7: MP #11, a training set: "three (upper left) vs. five" (upper right)"

Maksimov interpreted the question posed by Bongard not as, "What are the cognitive mechanisms that underlie the solution of BP's by people?", but as, "Is it possible to write a program that can perform as well as people?" Accordingly, he employed what was considered standard AI philosophy at the time, treating every problem of cognition as one of a *search* in a combinatorial space of states, and applying various heuristics to curb the exponential growth and reach a goal state within a reasonable time. He achieved this as follows.

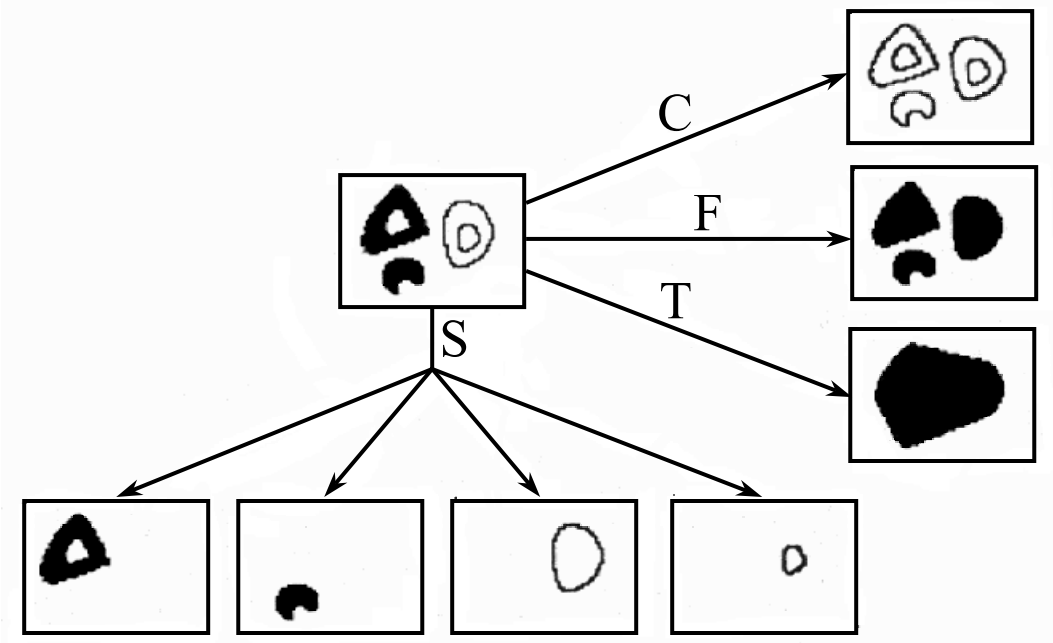


Figure 4.8: Tree of descendant images, applying operators *contour isolation* (C), *contour filling* (F), *convex hull filling* (T), and *separation by connectedness* (S)

A number of operators were applied to a given input image, resulting in a corresponding number of descendant images (Figure 4.8). For example, given an image with an irregular filled figure as input, one operator would create a new image with the convex hull of the figure; another operator would find the outline of the filled figure and create a new image with merely this outline in it; and so

on. Each of these new images was a descendant of the initial one. Then the operators were applied recursively to the descendants, thus producing further images and expanding the tree at deeper levels. There were other operators that could extract numerical data and make binary (i.e., Boolean) decisions. For example, one operator computed the area of a shape (the number of its pixels); another (Boolean) operator would decide whether a shape was outlined or filled; and so on. The search tree continued being expanded until one of the operators resulted in a set of Boolean numbers that had the same value (e.g., *true*) for all images on the left side, and the opposite value (e.g., *false*) for all images on the right side.

One of the most crucial algorithms in this approach was the *grouping of data into lots*. By this, Maksimov meant a procedure applied to a collection of real numbers distributed randomly — but not uniformly — within a range, so that they form some natural (i.e., visually discernible) groups. The procedure determines the groups and assigns each number to a group. The algorithm proceeds by assuming a number of groups, k, into which the data must be divided (i.e., assuming a value for one of the parameters that must be determined) and computing a number κ, called the *index of compactness,* from k and the data. It then tries to minimize κ by choosing different values for k, because the lowest index of compactness turns out to be the one for which the data are "best" grouped into lots. Since it would be computationally prohibitive to try *all* values of k in order to choose the best resulting κ, the algorithm makes an *ad hoc* decision, stopping at whatever κ is first found to be smaller than a pre-determined threshold, κ^*. For example, it is reported that the best value for κ^* was found experimentally to be 0.25 (for all cases of group-formation, in all MP's).

This procedure, which was Maksimov's solution for the AI problem of *group formation,* or *classification,* can itself be classified into one of the well-known families of algorithms popular in the field of *data mining* (Jain, Murty *et al.*, 1999). [10] Such algorithms can be useful for computationally intensive grouping of data, as is common in some engineering fields of AI, where the emphasis is on the result, not on the method used to reach it. On the contrary, in psychology, in which it is the *method* that is investigated and modeled, the criterion for correctness is compatibility with human cognition.

Maksimov seems to claim (in the description of his system) that his approach is psychologically plausible *because* there is agreement with human responses (though we are not informed about the nature and timings of such responses). Two factors, however, diminish the plausibility of his assertion.

First, many of his MP's are too "machine-oriented" — that is, their solutions are such that they can be expressed in terms of primitives available to Maksimov's system (e.g., "length of curved lines") or are reachable by tree-search (e.g., "the length of the contour of the convex hull", as in MP's #23 and #24), but hardly appear natural to people.

Second, Maksimov's program operated by selective "brain surgery": because there were severe restrictions on the amount of data that could be present at a given time in memory, some pieces of code that were deemed irrelevant for certain MP's were removed when the program was set to work on those MP's — or else the program and the data could not be present simultaneously in memory. The trouble is, however, that when a human operator determines which pieces of code will or will not be used in a given problem, the system becomes semi- rather

[10] The use of a threshold suggests that Maksimov's algorithm is a variant of the *k*-neighborhood algorithm (see Jain, Murty *et al.*, 1999).

than fully automated. It is unknown what the system's performance would be if the program were present in its entirety at all times, and were left to operate on its own devices.

In summary, although Maksimov's program constitutes the most thorough and sincere computational effort made so far in the domain of Bongard problems, it falls far short of proposing an adequate, and — most important — cognitively interesting approach.

4.3 How is Phaeaco's approach different?

Unlike Maksimov's system, Phaeaco uses the original BP's as input (plus those designed later by Hofstadter and the author, all listed in Appendix A). And, unlike RF4, Phaeaco uses black-and-white pixels as the form in which input is encoded. Each BP in Phaeaco's input consists of 12 boxes, presorted into two groups (left and right), and each box has a resolution of 100 × 100 pixels. Phaeaco "looks" at the pixels in the 12 boxes, initially in parallel, and then concentrates increasingly on particular boxes, according to the visual patterns perceived in them. The details of Phaeaco's processing of input will be explained in chapter 10.

Some readers might object to the idea that examining black-and-white pixels at a resolution of 100 × 100 constitutes "image-processing" in any significant sense.[11] Such readers would require true-color photographs as input before applying the label "image-processing" to algorithms.

The answer to this concern is twofold. First, Phaeaco is not limited to black-and-white pixels. It can accept true-color photographs as input, in which case it applies traditional contrast-enhancement and edge-detection filtering methods to

[11] Many thanks to Katy Börner (personal communication) for raising this issue.

convert the input to a black-and-white representation (more in §10.1). And second, even with a resolution of 100 × 100 (although Phaeaco is by no means *limited* by this value) there is plenty of room for ambiguity in the input. Even though individual pixels have sharply defined values (**0** or **1**), *collections* of pixels are almost never sharply defined as objects. For example, when is a hand-drawn collection of pixels a piece of a straight line, and not part of a curve? When is a polygon a circle, and a rectangle a square? Is a hand-drawn scanned figure really open, or should it be perceived as closed in the context of other similarly hand-drawn closed figures? To answer these questions, and many more, Phaeaco does not employ crystal-clear definitions, but relies on context. All these potential sources of ambiguity, together with a non-rigid way of looking at the input, make it much easier to claim that Phaeaco performs not simply traditional image-processing, but *cognitively interesting* processing of the input, starting at a very low (raw) level.

Image-processing is not Phaeaco's focus, however, but only a means of ensuring that the input is not formalized and preprocessed, as is the case with systems such as RF4. Phaeaco proceeds beyond image-processing, to visual pattern-formation, storage of patterns in LTM, priming of related patterns through a mechanism of spreading activation (not unlike mechanisms in artificial neural networks), pattern-matching, and recall from LTM. This is the "cognitive processing" of Phaeaco, in which there is no tree-like search in a combinatorially growing space. Visual patterns, for example, grow to some extent due to the action of small pieces of code, called *codelets,* that act on patterns and compete against each other, vying for computing time. These characteristics have been borrowed from the Copycat family of architectures (Hofstadter, 1995a; Marshall,

1999; Mitchell, 1993; Rehling, 2001) — of which Phaeaco is another member — and will be explained in more detail in §6.2.

Thus, processing in Phaeaco occurs at two levels that interact with each other. The first, or "retinal", level, is more akin to the processing of visual input in the retina and visual cortex, although it does *not* attempt to model those brain modules at the neurophysiological level. The second, or "cognitive", level, comes closer to modeling human psychological processing, and is the level at which a scheme of conceptual representation is employed. The two levels interact and influence each other, working in parallel. The retinal level starts working first, since it processes the raw (pixel-based) visual input. The cognitive level begins as soon as possible, in a pipelined fashion, and can modify certain parameters that modulate the functioning of the retinal level. This top-down (cognitive-to-retinal) flow of information is limited, however; the bulk of information-flow occurs in the bottom-up direction (retinal-to-cognitive).

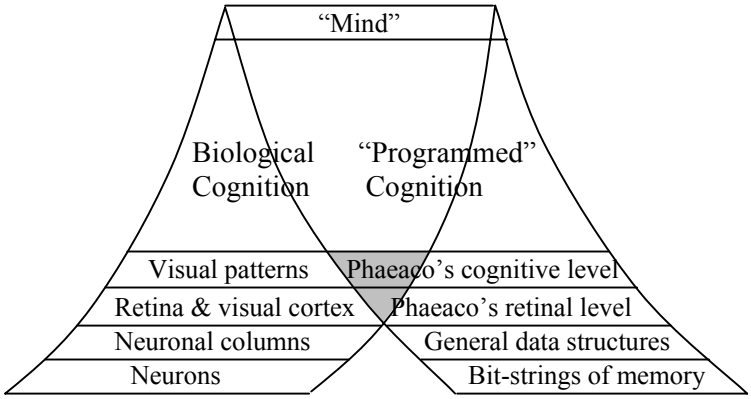


Figure 4.9: Expectation of convergence between biological and "programmed" cognition

The reason for distinguishing between retinal (lower) and cognitive (higher) levels of organization is that biological and "programmed" cognition[12] operate on different hardware components. The former uses neurons as its lowest-level hardware building blocks, and the latter uses bit-strings of memory (Figure 4.9). Some designers of cognitive systems (connectionists) simulate neurons through bit-strings. In Phaeaco, neurons are considered to be inappropriate targets of simulation. Instead, biological and programmed hardware components are seen as initially independent; but through successive levels of abstraction in each case, they take on almost identical functions. The similarity increases in proportion with the degree of abstraction. Eventually, it is hoped that at the highest level of abstraction, we reach what we experience as a "mind".

Thus, the answer to the question "What does Phaeaco model?" can be found by looking at the intersection of cognitive and programmed cognition in Figure 4.9: Phaeaco models *some* fundamental cognitive processes, and the number of human-like cognitive behaviors is expected to be larger as we move higher up in the abstraction hierarchy toward a fully developed "mind" (the shared area in Figure 4.9 is larger at the cognitive than at the retinal level).

---

[12] The term "programmed cognition" refers to any attempt to implement models of human cognition in computers. The expression "artificial intelligence" is avoided here because the term "AI" has acquired distinctly engineering and even science-fiction connotations in recent years, distancing it from the goal of understanding how the mind works.

4.4 What should be the goal of automation?

Given the earlier attempts at automating the BP-solving process, a question that naturally arises is, "How should one compare various approaches?" Since the question involves comparisons of multidimensional systems, there can be no simple answer. The following subsections examine various dimensions of BP-solving systems and the corresponding space and "metrics" that arise from them.

4.4.1 Number of BP's solved vs. degree of automation

Is it correct to look exclusively at the number of BP's solved by some system in order to form an opinion on how good, or satisfying, the approach is? It might seem that the answer to this question is subjective, and that it all depends on how one defines a "good" or "satisfying" approach. However, to make it clear that the idea of looking *only* at the number of BP's solved (as a percentage of the original 100, for example) is not a good metric of "goodness", consider the following extreme case.

Imagine a "program" that consists of an array of 100 strings, the *n*-th string of which contains the solution of the *n*-th BP (in plain English) in Bongard's original collection. Given one of the 100 BP's as input, the problem is first encoded as a number (by a human helper), and the number is then given to the "program", which outputs the *n*-th string.

What is wrong with this approach? An immediate concern is that it is *not productive*: it can solve only those BP's that are already stored in its table. But an extension to this approach can have the human helper adding strings to the table as each new BP appears, and since it is up to this helper to decide a number given

the BP, every BP that has been seen at least once can be solved in any of its future appearances.

Thus there is no problem with the productivity of this approach, but there is a problem with its *automation*. The problem is that the human helper, not the program, solves the BP. As an automation method, this would be absolutely unacceptable, but it serves to remind us that simply counting the BP's solved is an unacceptable metric of the "goodness" of an approach. It may be of some use to consider the number of solved problems, but the degree of automation (independence from human help) in a system must also be considered.

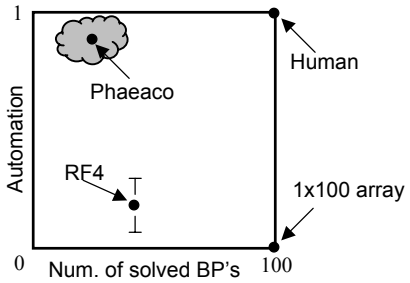


Figure 4.10: Graph of number of solved BP's vs. degree of automation

The graph in Figure 4.10 depicts the space of "number of solved BP's" vs. "degree of automation". Phaeaco has been placed on this space somewhere with a relatively small number of solved BP's (at its current stage of implementation), but with high automation value.[13] RF4 is also shown, with 43 solved BP's and a low automation value, since a human helper must convert the BP's into first-order logic formulas. The cases of a human being and the trivial "program" with the 1×1000 array are also shown at the extremities. Maksimov's system is not shown,

[13] This value is not exactly maximum because Phaeaco's current implementation does not include a camera to look at a BP, so some human helper has to scan the BP first and create an image file in a format that Phaeaco can read; alternatively, a BP can be created directly using Phaeaco's bitmap editor.

since it is not known to have solved any of the 100 BP's, but if it were depicted on the graph, its automation value would be comparable to Phaeaco's.

4.4.2 Agreement with data vs. interest in cognitive science

At least two more dimensions of automated approaches are interesting. One is how closely the behavior of a system agrees with measurements of human behavior. Maksimov, for example, claimed a significant degree of agreement between his system and data collected from human subjects, although he did not quantify this agreement. But even a system having 100% agreement with human behavior is not necessarily cognitively interesting. An example of this case — not from the BP domain — is Deep Blue, the computer chess program that, in 1997, became the first program ever to defeat in a match the reigning world chess champion (Hsu, 2002). Though Deep Blue's performance was such that it could possibly pass a chess-restricted "imitation game",[14] its approach for achieving its goal (which was simply to defeat the world chess champion) was to employ state-of-the-art heuristics for searching and pruning the exponentially growing tree of chess moves, and even more to rely on very fast computing hardware. Thus, Deep Blue made no attempt to model any aspect of human cognition.[15] From a cognitive science perspective, the only lesson learned from Deep Blue's victory

---

[14] This is the term Turing used for what became known as "Turing Test". Since the Turing Test is by its definition unrestricted, one might consider an "imitation game" restricted in the domain of chess, where a human judge tries to understand whether the chess-playing opponent is human. This is an excerpt from Hsu's book: "Somehow, all the work caused Grandmaster Joel Benjamin […] to say, 'You know, sometimes Deep Blue plays chess.' Joel could no longer distinguish with certainty Deep Blue's moves from the moves played by the top Grandmasters." (Hsu, 2002).

[15] Hsu writes in the preface of his book, "We approached the problem [of computer chess] from a different direction. We, or at least I, viewed the problem as a purely engineering one. […] Our project began with a simple goal, namely, to find out whether a massive increase in hardware speed would be sufficient to 'solve' the Computer Chess Problem." (Hsu, 2002).

against the world chess champion was that there are alternative (non-cognitive) computational ways to emulate some aspects of cognition.

If we were to construct a graph with two axes, "agreement with human behavior" vs. "cognitive interest", and plot the locations of Phaeaco and Deep Blue, we would arguably obtain something like the graph in Figure 4.11.

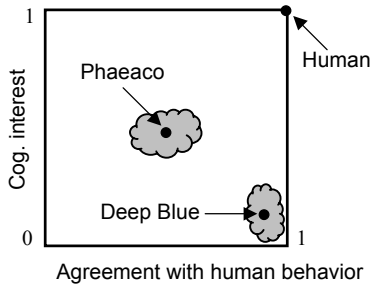


Figure 4.11: Graph of agreement with human behavior vs. cognitive interest

Note that the estimate of "agreement with human behavior" for Deep Blue in Figure 4.11 is based on expert opinion (Hsu, 2002, see also the footnotes on the previous page). Less sophisticated judges might express an even stronger conviction that there is human intelligence behind Deep Blue's performance.

In conclusion, just as examining a person's ability in a single skill is usually considered inadequate as an assessment of a person's overall intelligence, so a single measurement along any of the dimensions discussed above is inappropriate as an estimate of the success of a BP-solving approach. Nonetheless, analogously to the single value obtained by an IQ test for a person, one can envisage using some ordinary metric in a multi-dimensional space to compute a "distance from human" as a single-valued estimate of the "goodness" of an automated approach.

PART II: Phaeaco

CHAPTER FIVE

Phaeaco in Action

Phaeaco's external appearance and behavior when asked to solve BP's is presented in this chapter. The captured images of the program show not merely Bongard problems but Phaeaco's entire interface after some processing has occurred; in addition, the solution to the problem has been printed at the bottom of the screen. It should be noted that the resolution of the input that Phaeaco processes is exactly as shown in the various BP's printed throughout chapters 1–4, in which the individual pixels are easily discernible, and not as shown in the figures of the present chapter, in which the dimensions of BP's are reduced to make room for the display of the program's entire interface.

5.1 What Phaeaco can do

In the subsections that follow, the BP's that Phaeaco's current implementation can solve are grouped into categories, according to the deeper issues that the architecture must manage. The way these issues are handled is explained in subsequent chapters. This chapter provides a general preview only.

5.1.1 Feature value distinction

The solutions of an estimated 44% of all BP's are based on distinguishing between the values that some feature has on the left and right sides. For example,

the feature could be the size (area) of an object, and the distinction between values could be "large vs. small" (as in BP #2).

The problem that Phaeaco solves faster than any other is BP #3 (shown in full resolution in Figure 1.2), which has a solution based on a discrete-valued feature.

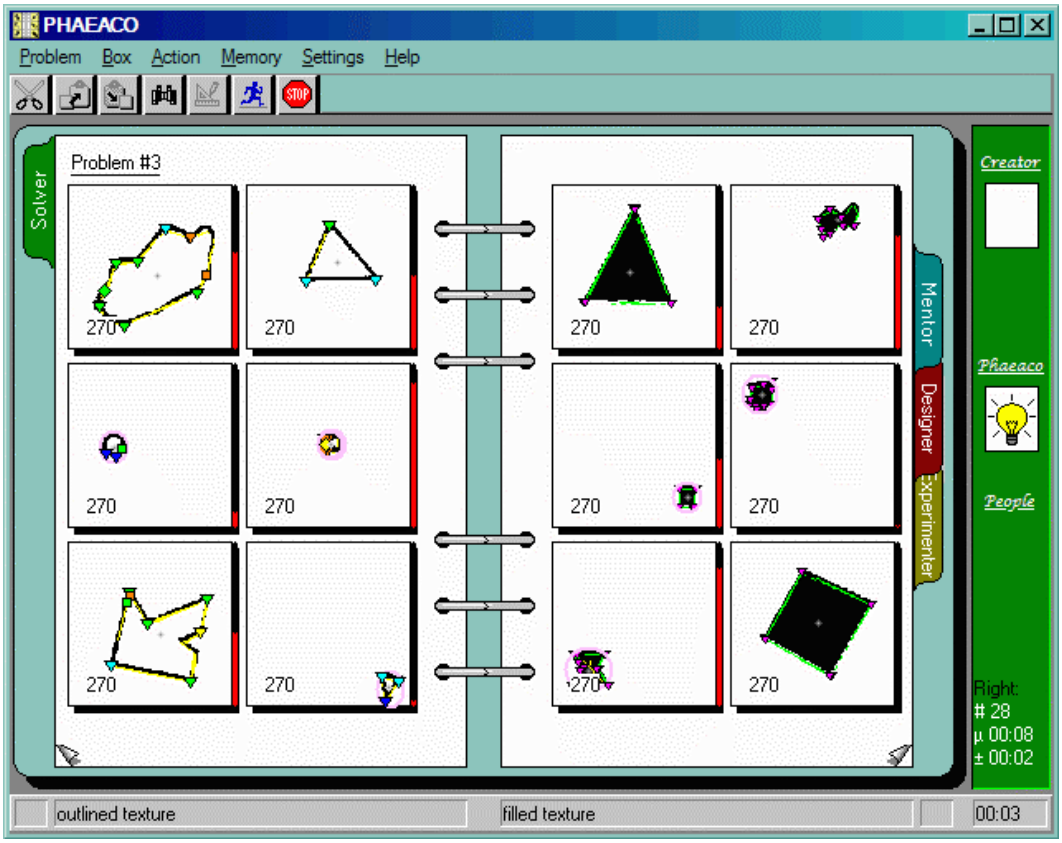


Figure 5.1: Phaeaco's appearance after having solved BP #3

Figure 5.1 shows Phaeaco's interface after having solved BP #3. The answer reached by the program is printed at the bottom of the left and right pages ("outlined texture" and "filled texture"). The time, in seconds, taken to solve this BP is shown at the rightmost corner of the display ("00:03"). Because such times are largely dependent on the speed of the computer, only relative evaluations of

them are meaningful. Phaeaco solved BP #3 on 98% of its attempts, as compared with 28 human subjects, all of whom solved the problem, with an average time of 7.9 seconds and a standard deviation of 6.1 seconds.

The marks on the vertices and along the sides of objects in Figure 5.1 are the result of animating the progress of some retinal-level processes, and will be explained in chapter 10. The number at the bottom-left corner of each box ("270") is the value of an internal clock-cycle counter, roughly corresponding to the time spent looking at the box. Other features of this interface will be discussed later in this chapter (§5.2).

In attempting to solve a BP, Phaeaco first goes through a "holistic stage", during which all 12 boxes of the input are examined simultaneously, devoting equal time to each box. BP #3 is one of a few problems that Phaeaco manages to solve without going into the next, "analytic" stage, in which attention shifts to individual boxes. (The details of the functioning of the holistic and analytic stages are given in §11.1.) Phaeaco reaches the solution by forming two visual patterns (§1.2.3), one for each side of the problem, and contrasting the two patterns in search of differences. In the case of BP #3, the pattern on the left almost always has the value "outlined" assigned to its feature "texture"; similarly, the pattern on the right has almost always a "filled" texture. The algorithm that contrasts the two patterns spots this difference immediately (due to a built-in high significance of the percept of texture[16]), and this signals the end of the solution-seeking process. The formation of an overall pattern for each side of the BP out of individual patterns for each of the six boxes is discussed in §11.1.2.

---

[16] This property is inspired by, but does not simulate, the ability of the human brain to spot contrast in colors immediately through color-specializing area V4 of the visual cortex (Zeki, 1993).

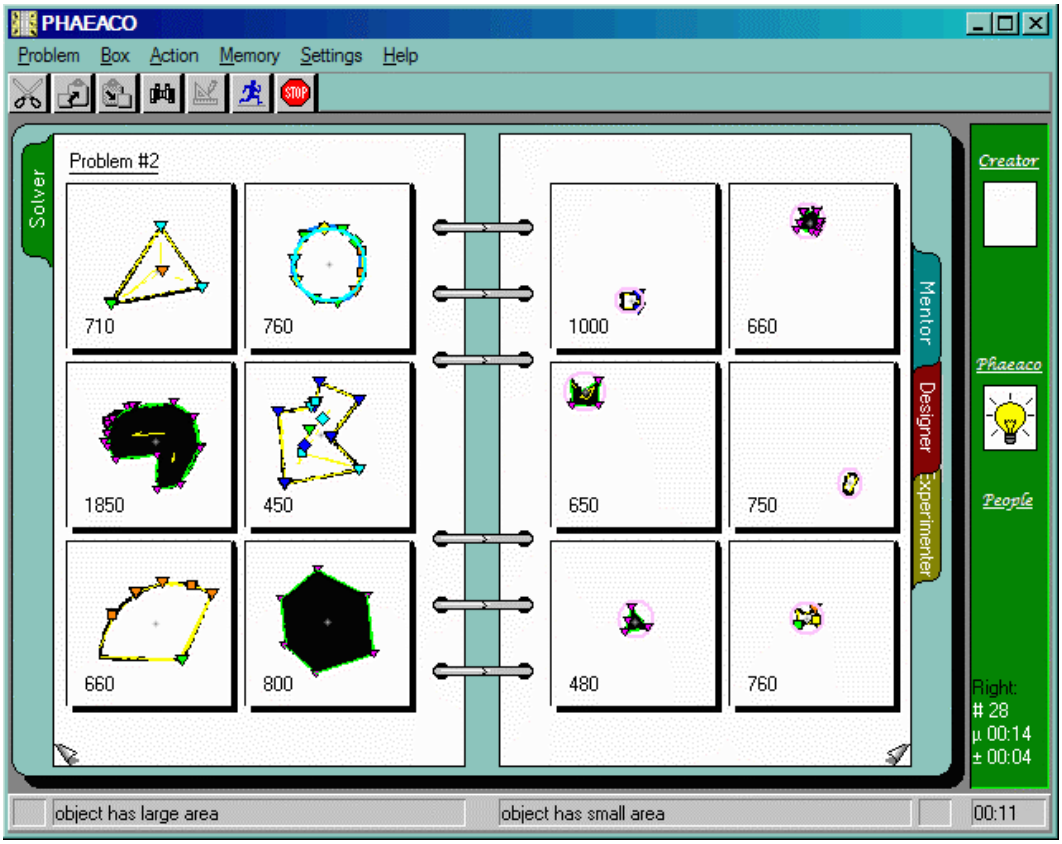


Figure 5.2: Phaeaco after having solved BP #2

When the feature values come from a continuous range, as in the case of "area" in BP #2 (Figure 5.2), Phaeaco compares the statistics of the values on the left against the statistics of the values on the right. For example, in BP #2, the areas of the six figures on the left side form a sample of size six that has a mean and a standard deviation. This sample, which is part of the left-side pattern, is compared statistically with the corresponding sample of the right-side pattern. If, through the use of well-known statistical methods, the two statistics are found to differ significantly, this gives a "subcognitive hint" to Phaeaco that the solution might be based on different areas. Phaeaco then verifies this idea by quickly

looking at each box and confirming that each figure on the left is larger than the largest figure on the right. Other problems solved in this manner are BP #8 and BP #11 (see Appendix A).

BP's with solutions based on numerosity (§1.2.1, and §7.3) are also solved by Phaeaco. One of the most straightforward is shown in Figure 5.3.

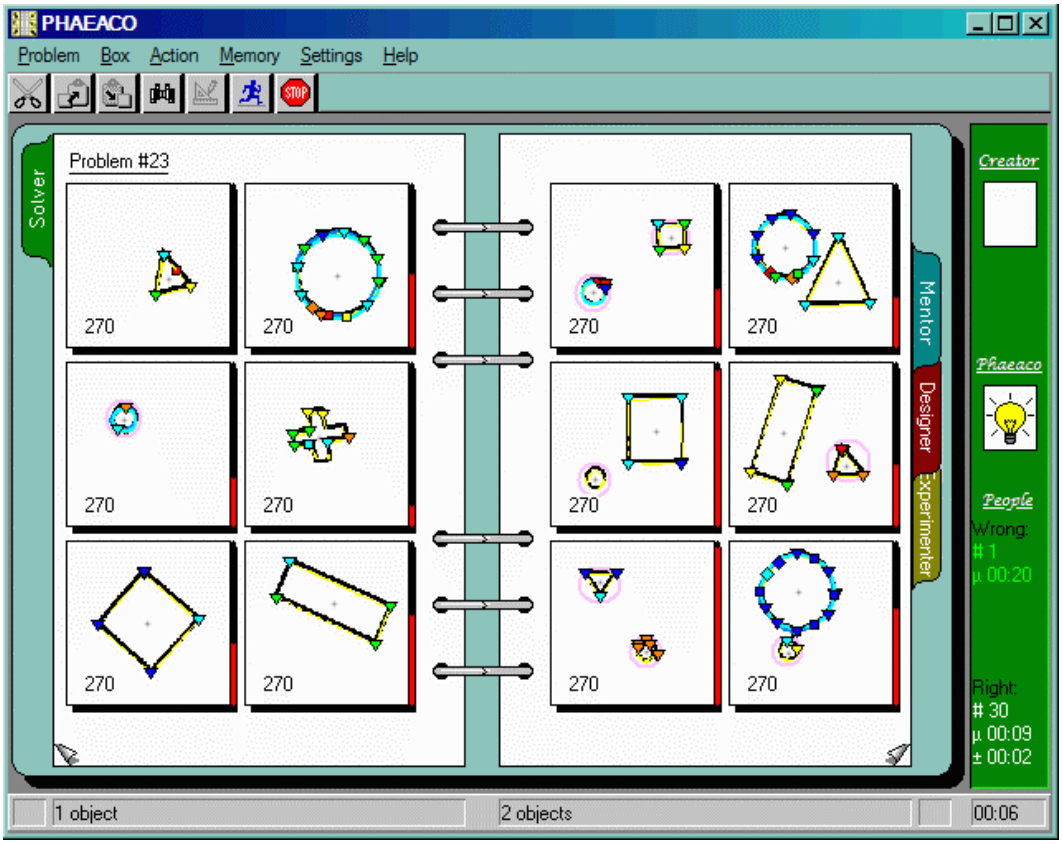


Figure 5.3: BP #23 as solved by Phaeaco

BP #23 is simple enough to be solved by Phaeaco by contrasting the numerosity of objects between the left- and right-side patterns. Other problems, however, such as BP #85 (mentioned in §4.1; see Figure 4.3), are more subtle.

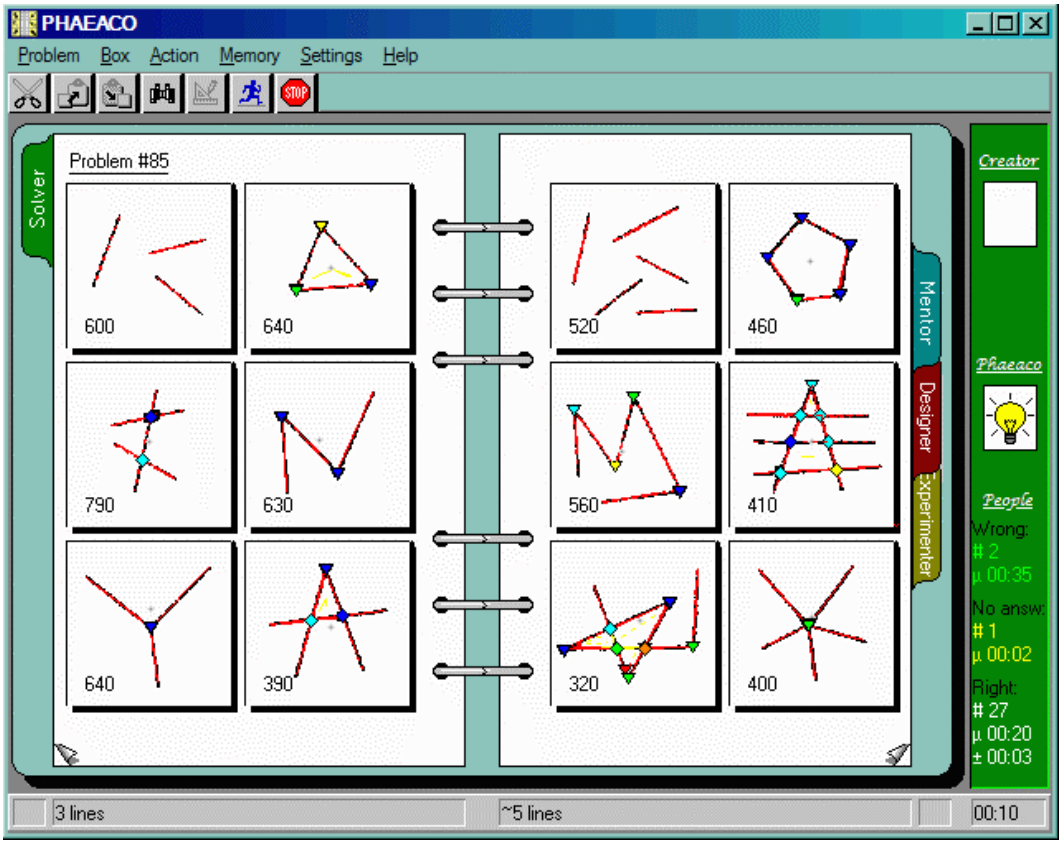


Figure 5.4: Difference in numerosity of lines in BP #85

The pattern that Phaeaco constructs on the left side of BP #85 (Figure 5.4) is not precisely "three lines". The reason is that only one box (I-A) depicts precisely three lines. Other boxes depict some structures (in particular, one of them is a triangle), and thus Phaeaco does not construct a pattern which is as simple as "three lines" for the left side, but rather one that can be described as "object with some lines", for five of the boxes; a similar description can be made for the right side. The program must spend additional time looking at the boxes individually (this is the analytic stage, mentioned above) before hitting upon the idea of differing numerosity of lines. Luckily, due to the simplicity of the input, this idea

usually appears very soon; in fact, it fails to appear in only 6% of attempts, as shown in the performance statistics in Appendix A. Interestingly, 10% of the subjects (3 out of 30) that attempted to solve BP #85 also failed to solve it.

5.1.2 Existence

BP #1 (Figure 5.5) is probably the simplest BP regarding existence, solved by 100% of human subjects and 100% of Phaeaco's attempts.

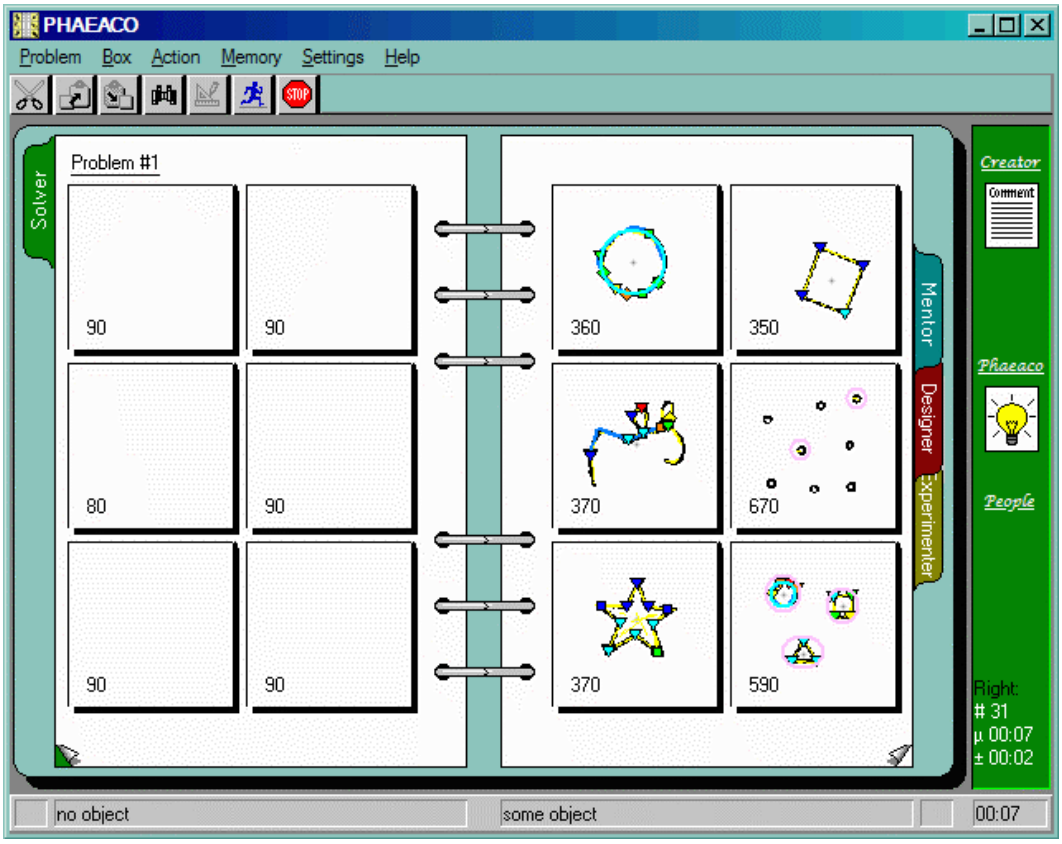


Figure 5.5: BP #1, the simplest problem of existence

Phaeaco spends nearly no time at all looking at the empty boxes of the left side of BP #1, and constructs an overall pattern for the left side that contains

nothing but a "box". The pattern of the right side, however, contains the notion "object", together with its numerosity (approximately equal to the average number of objects in a right-side box). Comparing these two patterns, Phaeaco immediately identifies the existence of a representational element that stands for "object" in the right-side pattern, and its absence in the left-side pattern.

Figure 5.6 shows BP #5, another problem solved by Phaeaco, the solution of which is based on the existence of curves (on the right side).

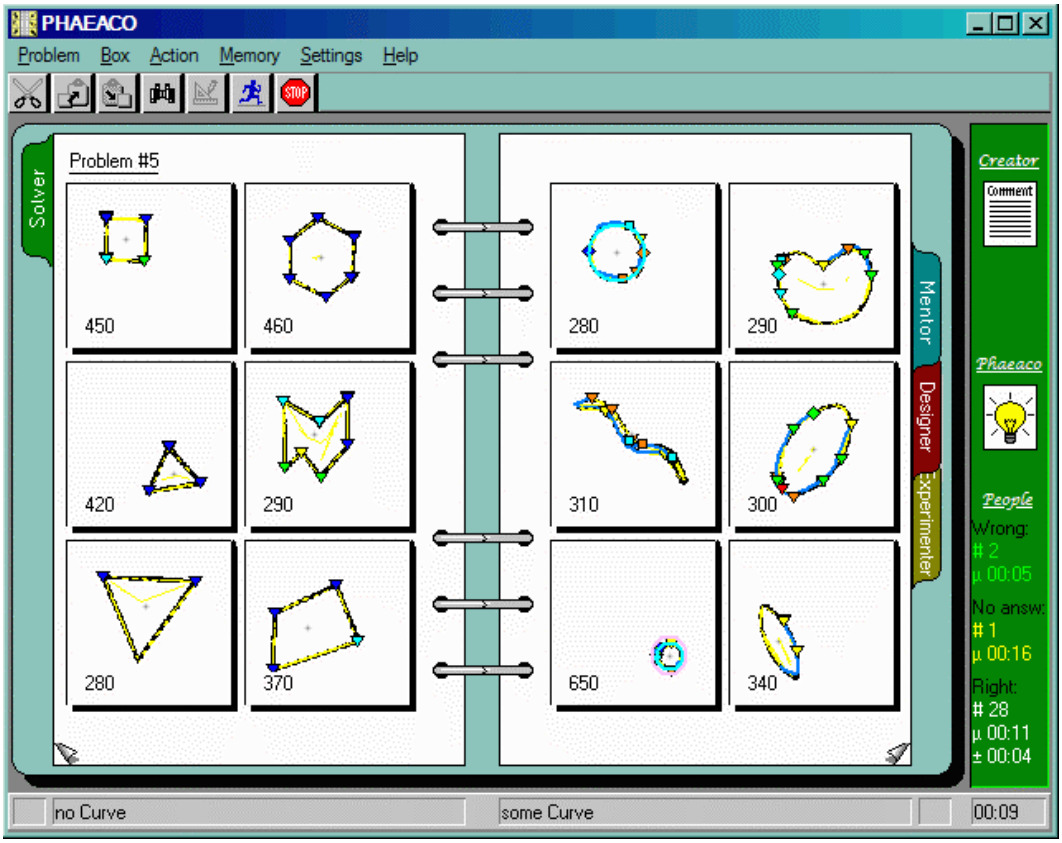


Figure 5.6: Existence of curves on the right side of BP #5

5.1.3 Imaginary percepts

Not all percepts that are necessary for the solution of BP's are always present explicitly in the input. One such example is the percept of the "convex hull" of a figure, which is the central idea in the solution of BP #4.

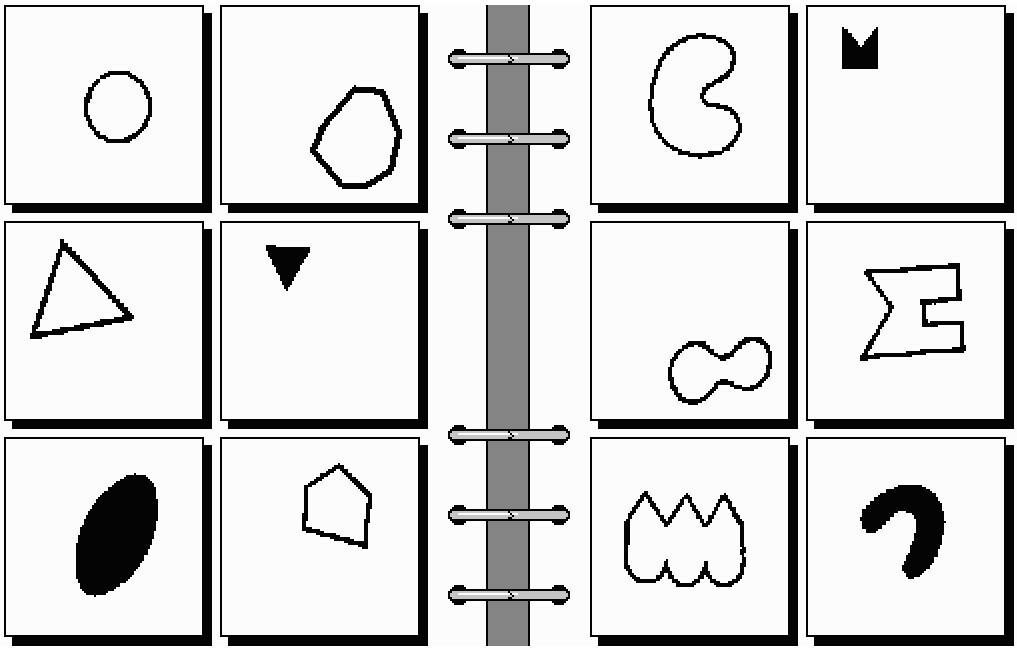


Figure 5.7: BP #4, "convex vs. concave"

In BP #4 (Figure 5.7), the left-side figures have no indentations (they are "convex"), whereas the right-side figures have at least one indentation (they are "concave"). One way to detect the existence of indentations is to imagine an elastic band enclosing the figure tightly. The elastic band forms a convex shape, the "convex hull" of the figure. The "difference" between the convex hull and the actual figure (i.e., those points that belong to the convex hull but not to the figure) reveals precisely the indentations of the figure.

Phaeaco has the ability to "imagine" the convex hull of a figure, and then to consider what remains if the figure is "subtracted" from the convex hull. Although imagining the convex hull is one of Phaeaco's "primitives" (in the sense that it does not depend on anything more fundamental), the operation itself is not performed immediately upon seeing a figure, nor is it at all certain that it will ever be performed. Perceiving the convex hull is a rather infrequent operation, and for this reason Phaeaco has a low success rate while attempting to solve BP #4 (20%). The success rate for human subjects is slightly lower, only 16%.

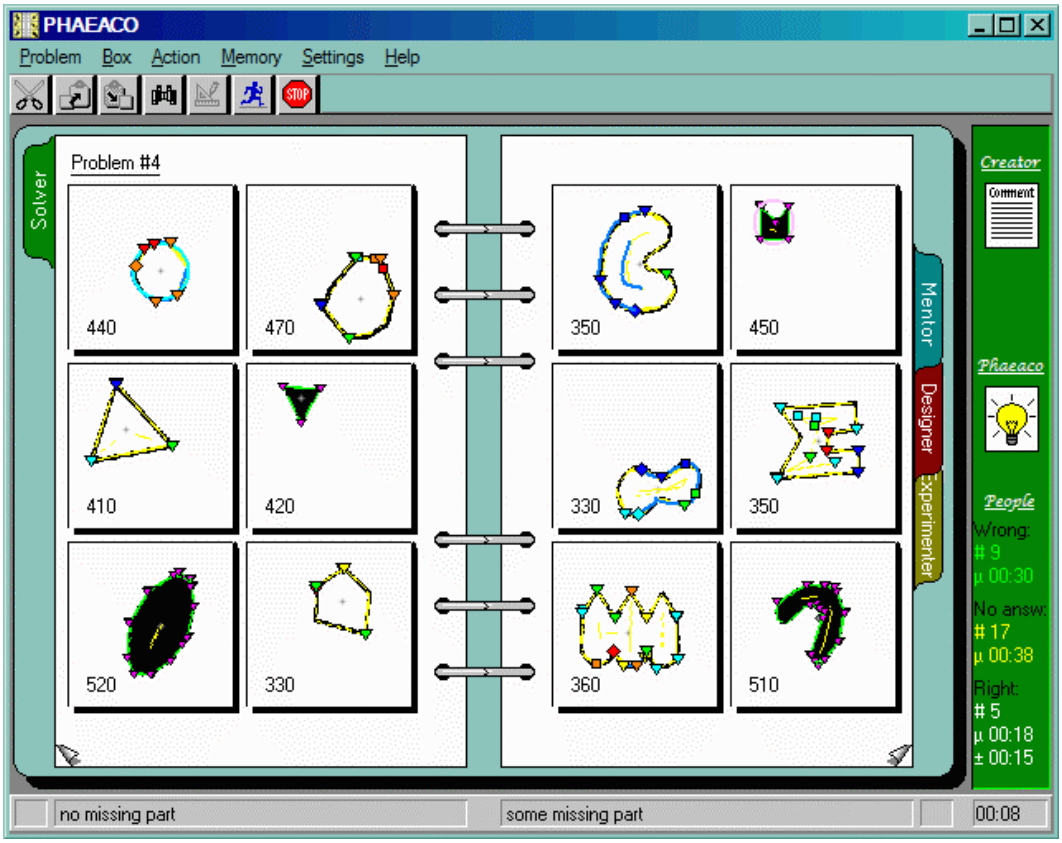


Figure 5.8: BP #4, as solved by Phaeaco

Note, however, that whenever close matches between Phaeaco and human subjects are reported, either in success rates or in relative response times, they are not meant to support the claim that human cognition is based on an architecture similar to that of Phaeaco's. Instead, close matches should be seen exactly for what they are: indications of the presence of human-like behavior on the part of Phaeaco.

Other perceptual elements that Phaeaco is in a position to "imagine" are straight-line segments and curves formed by the centers of small objects, and the interior of closed curves or polygons, which determine the solution of BP #15.

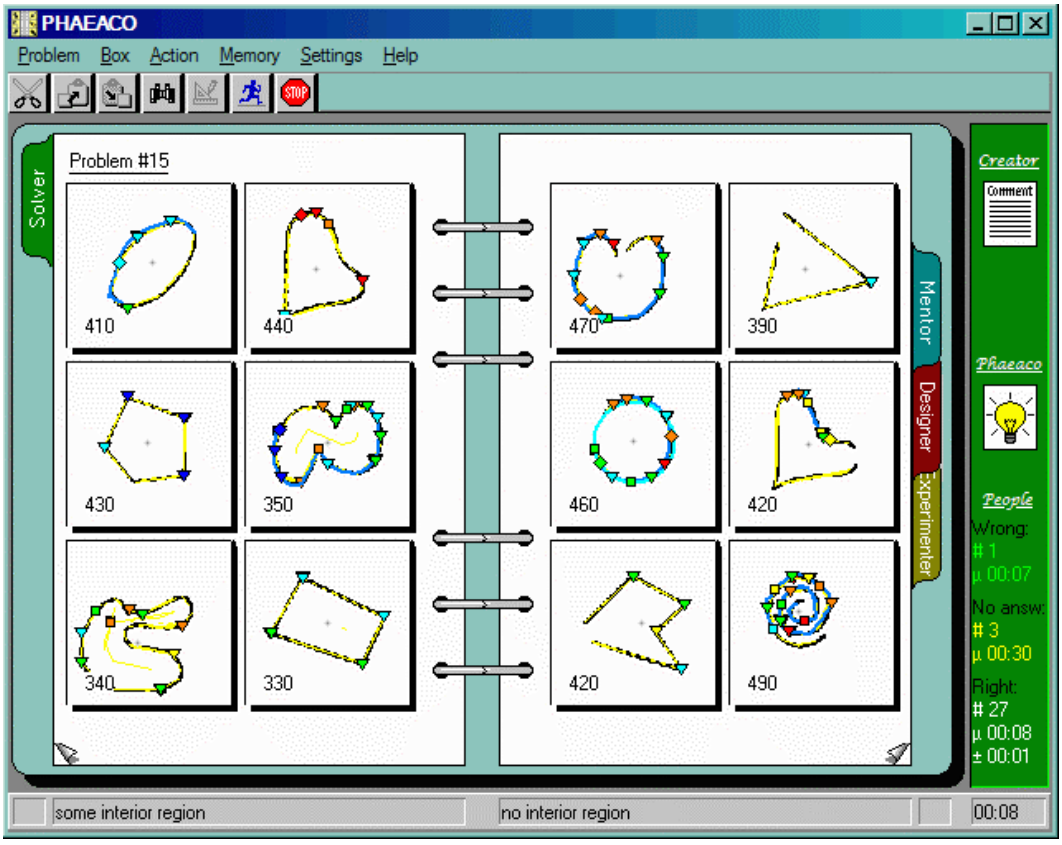


Figure 5.9: BP #15, as solved by Phaeaco

Interestingly, in its attempt to solve BP #15 (Figure 5.9), Phaeaco "imagines" a circle in box II-C, where only an incomplete circle exists. Similarly, in box II-B Phaeaco makes a very good match of the figure with the known pattern of a triangle, after extending the incomplete side and finding that it meets the endpoint of another side, thus completing the triangle. In spite of being able to construct imaginary closed figures, such as circles and triangles, Phaeaco is not distracted by them, noticing the existence of true closed regions only on the left, and their lack on the right.

5.1.4 Equality in a single box

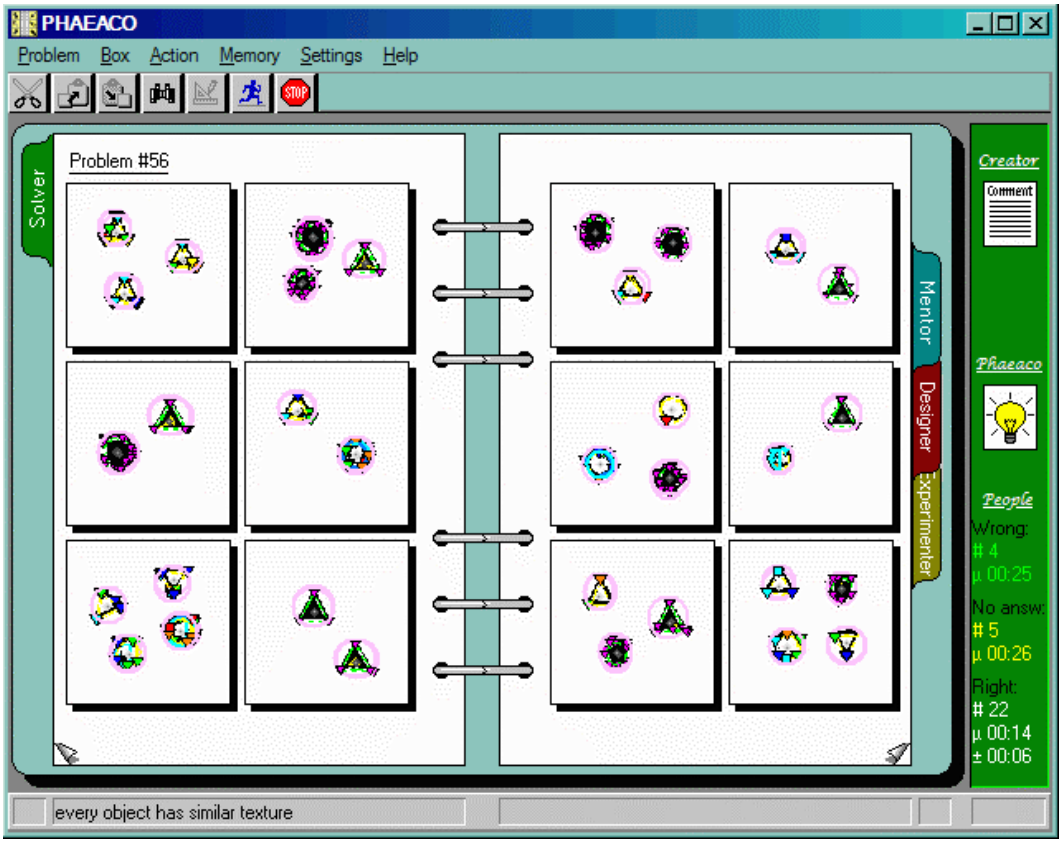


Figure 5.10: BP #56, as solved by Phaeaco

The solution of some BP's is based on a relation among features of objects that holds within each single box; though the features might differ from box to box, the relation remains the same across all boxes of a given side. An example of such a problem is BP #56 (Figure 5.10), for which the solution is that all objects within each box on the left are of the same texture. The texture can be sometimes "filled", sometimes "outlined", but whatever it happens to be, it is the same for all objects in a given box. On the right this is never the case.

This class of problems can be thought of as having solutions that are "one notch higher" in abstraction than all types of solutions we have seen so far. It is one idea to notice that "feature X of all objects in a box has the same specific value" (which can also be a legitimate BP), and another one to notice that "feature X of all objects in a box has the same value, *whatever that is*". Though on the surface the difference appears to be a minor one, the representation that needs to be built for Phaeaco to arrive at the latter, more abstract description, is quite different from the much simpler representation that suffices for the former type of solutions, as we shall see in §7.4.10.

Phaeaco does not solve BP #56 with ease, succeeding in only 20% of its attempts, failing to give an answer in another 64%, and giving a wrong answer in the remaining 16%. Human subjects, however, perform quite well on this problem, as seen in Figure 5.10: the percentage of successes is 71%, failures to answer are 16%, and wrong answers are 13%. This is a case where Phaeaco's scores of success and failure deviate substantially from those of human subjects. In truth, Phaeaco's 20% of successes can be improved substantially by some technical adjustments. Closer examination of Phaeaco's behavior shows that the correct answer ("every object has similar texture") is reached in many more attempts; but Phaeaco often fails to verify that it is true in all boxes on the left and

false in all boxes on the right. If even one of the 12 boxes does not pass the verification stage, the solution is rejected and a different one is sought. Phaeaco's vision sometimes is not as acute as necessary to see precisely what there is in each box, all the time. Since the disagreement with human performance became evident only after the experiment with human subjects was administered, no *a posteriori* corrective action was taken to make Phaeaco's performance match the human one better.

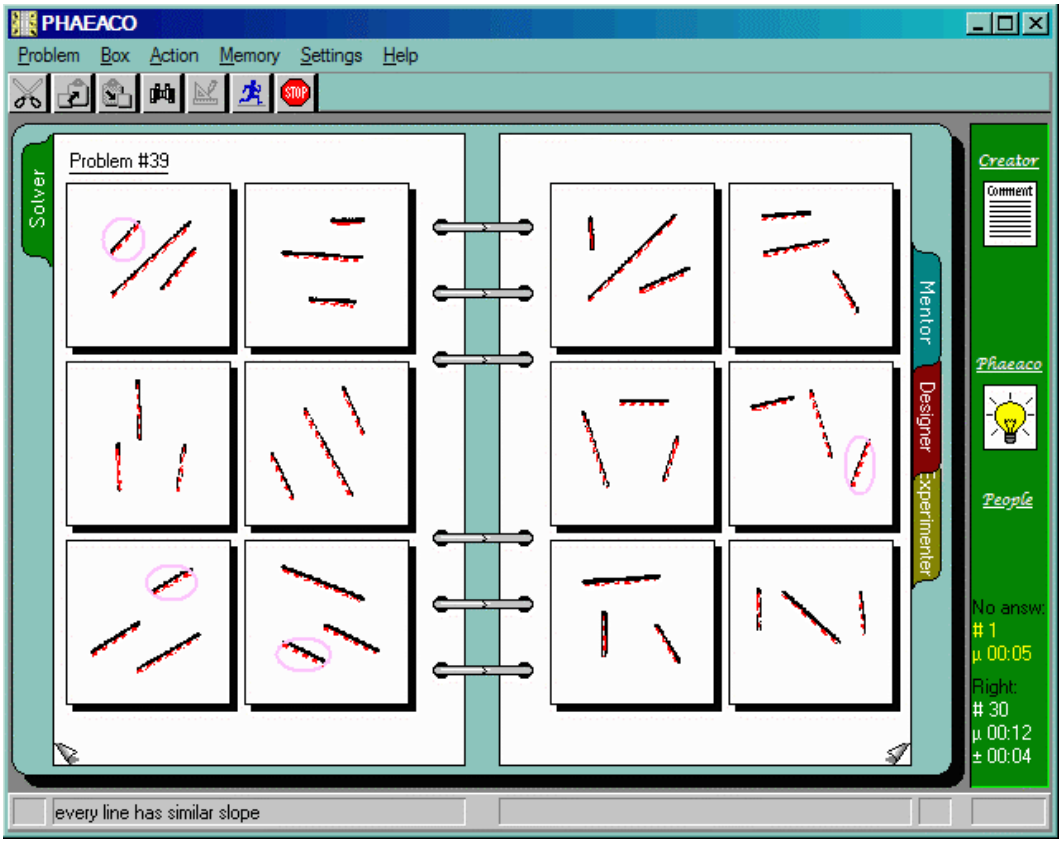


Figure 5.11: BP #39, as solved by Phaeaco

Figure 5.11, as well as Figure 5.12, shows problems of the "same kind" as BP #56. In BP #39 the feature that keeps approximately the same value across all

objects (line segments, in this case) in each box is slope; in BP #22, it is the area of objects (approximately equal, of course, since slope and area are continuous-valued features). These two problems are landmark cases in Phaeaco's development because their solutions were reached "for free", i.e., after Phaeaco becoming sophisticated enough to solve BP #56, it could solve BP's #39 and #22 without a single line of code having to be added.

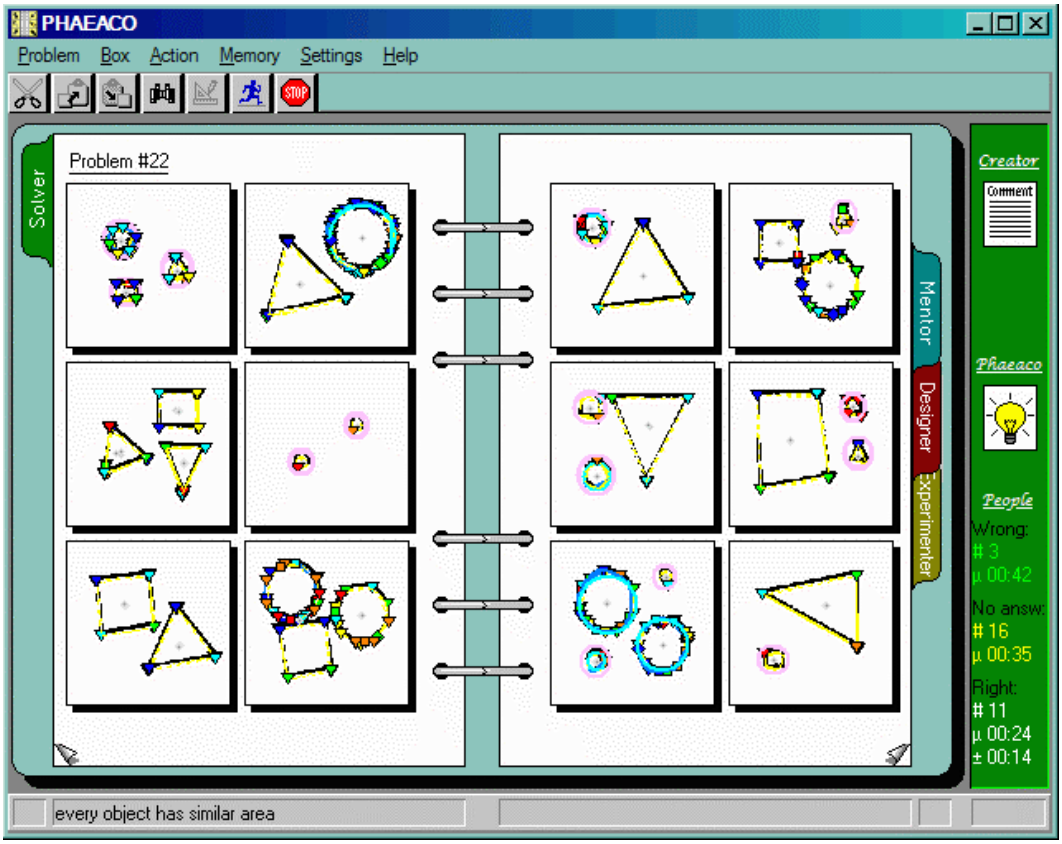


Figure 5.12: BP #22, as solved by Phaeaco

This is important, because ideally this situation should be repeated often. As the system becomes more robust, adding programming lines to deal with a specific problem should result in the solution of a number of other problems that

have solutions with similar characteristics. Eventually, a time should come when Phaeaco's programmer adds the last line of code in the implementation, and Phaeaco becomes capable of solving *every* BP that is also solvable by humans. Could this ever be true? This is an interesting philosophical question. If it were not true, then there would always be some BP's that are solvable by people but that lie forever beyond Phaeaco's cognitive horizon. Unless we are willing to assume that the human mind possesses some "magic" property that can never be programmed, the answer to the previous question must be affirmative: a time must come when the last added piece of programming code endows Phaeaco with just as much fluidity, abstraction ability, and creativity in coming up with an answer as the best of human minds.

Nonetheless, situations such as this one (i.e., an enhancement in Phaeaco's architecture resulting in BP's getting solved "for free") must be exceedingly rare in Bongard's collection of 100 problems. Certainly the case described above was the only one observed in the present implementation. The reason is that Bongard, being a good problem-designer, did not often repeat himself. He preferred to apply each abstract concept — such as "all objects have the same value for some feature" — in no more than two or three problems, frequently moving on to explore different ideas.[17] Yet, no matter how creative a designer is, there must be an end to creative ideas that can be expressed in 12 black-and-white, 100×100 boxes. Under these restrictions, the set of possible BP's is vast, but finite. And experience in BP-design shows that with more than 300 BP's collected from various sources,[18] ideas do start being repeated. In a rough, educated guess, a set of 10,000 BP's could include the ideas of all but the most creative designers.

---

[17] This is true to an even higher degree in Hofstadter's collection of 56 BP's.

[18] Mailed to the author by BP-enthusiasts through the internet.

5.1.5 Reading the small print

All the previous subsections described successive architectural enhancements that were implemented in Phaeaco in order to cope with increasingly complex issues. This subsection describes a technical enhancement without which even some of the already presented BP's would remain unsolvable.

The enhancement pertains to Phaeaco's visual perception of small objects. According to conventional wisdom in traditional image processing, if an object appears large (i.e., many pixels make up its image), it requires *more* processing, because there is more information that must be processed before a program can form an "opinion" on what it represents. This in turn entails more opportunities for a traditional program to be "lost in the details" of a large image, and arrive at the wrong conclusion. Smaller objects typically are easier to process. In Phaeaco, the opposite is true. Not all pixels of a large object need be explicitly processed, as will be explained in §10.3 before Phaeaco can "sense" what the object looks like (i.e., before it matches it to a known object). Thus, in Phaeaco, "smaller" rather than "larger" implies "harder to perceive", because the coarseness of resolution at very small sizes yields ambiguities. Consider Figure 5.13.

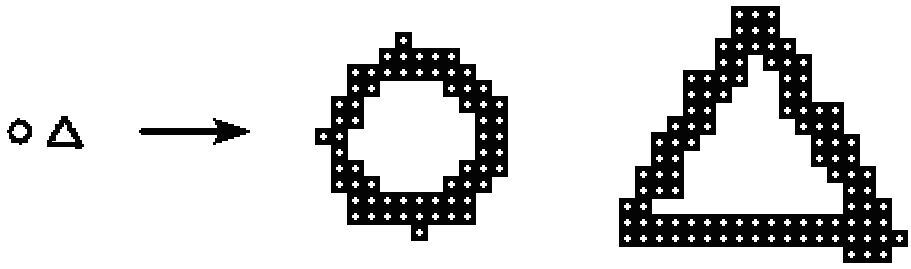


Figure 5.13: Left of arrow: two small objects. Right of arrow: their actual pixels magnified

Figure 5.13 shows on the left two small objects, a circle and a triangle, taken from the box of a BP. The right side of the same figure shows the magnified pixels of these two objects. While looking at the small objects, the human eye manages to fill in the missing pixels and to smooth the lines; as a result, the circle looks to us like a circle — it would not be confused with a polygon, for example. But when we look at the actual pixels on the right, we see what the program sees: the circle now could be construed as a polygon (a heptagon, perhaps), and the imperfections of the triangle appear exaggerated. If the two shapes were larger, the imperfections of their lines would average out; but at this resolution they could cause some confusion for the image-processing algorithms.

For such objects, Phaeaco has the equivalent of a "magnifying glass": a set of algorithms that magnify the objects by adding pixels appropriately, so that their imperfections are smoothed out. The following figure shows the result.

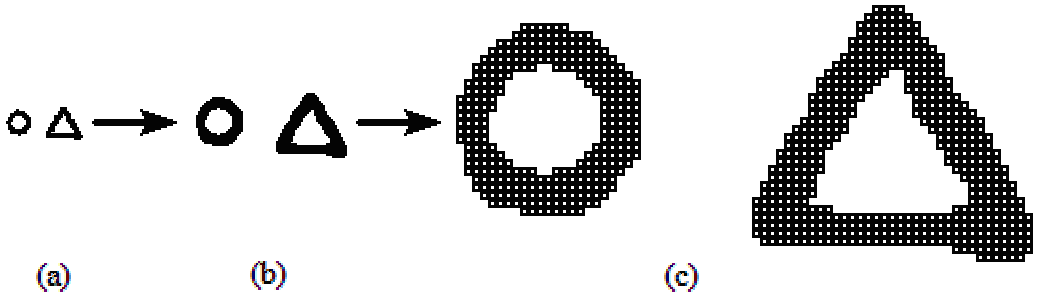


Figure 5.14: Result of algorithmic magnification (b) and actual pixels (c)

In Figure 5.14, the original small objects (a) are magnified and smoothed at the same time (b); the actual pixels of (b) are shown more clearly in (c). We see that now the pixels are more numerous than those of the objects in Figure 5.13, and the contours appear smoother not only to the human eye, but objectively (to the program). Any magnification size is possible by the algorithm that achieves

this result, as will be explained in detail in §10.3.17, but in practice a factor of two works well in most cases. In the same section, it will be explained that Phaeaco does not have a sharp but rather a probabilistic threshold for areas below which objects are considered "small": there is a gray region of area values in which there is some probability that Phaeaco will apply the magnification algorithm. The chances diminish sharply for larger objects, and increase sharply for smaller ones.

Figure 5.15 shows a problem in which applying the magnification algorithm is essential for Phaeaco's success. Small objects on which the magnification algorithm has been applied are depicted enclosed in a faint gray circle.

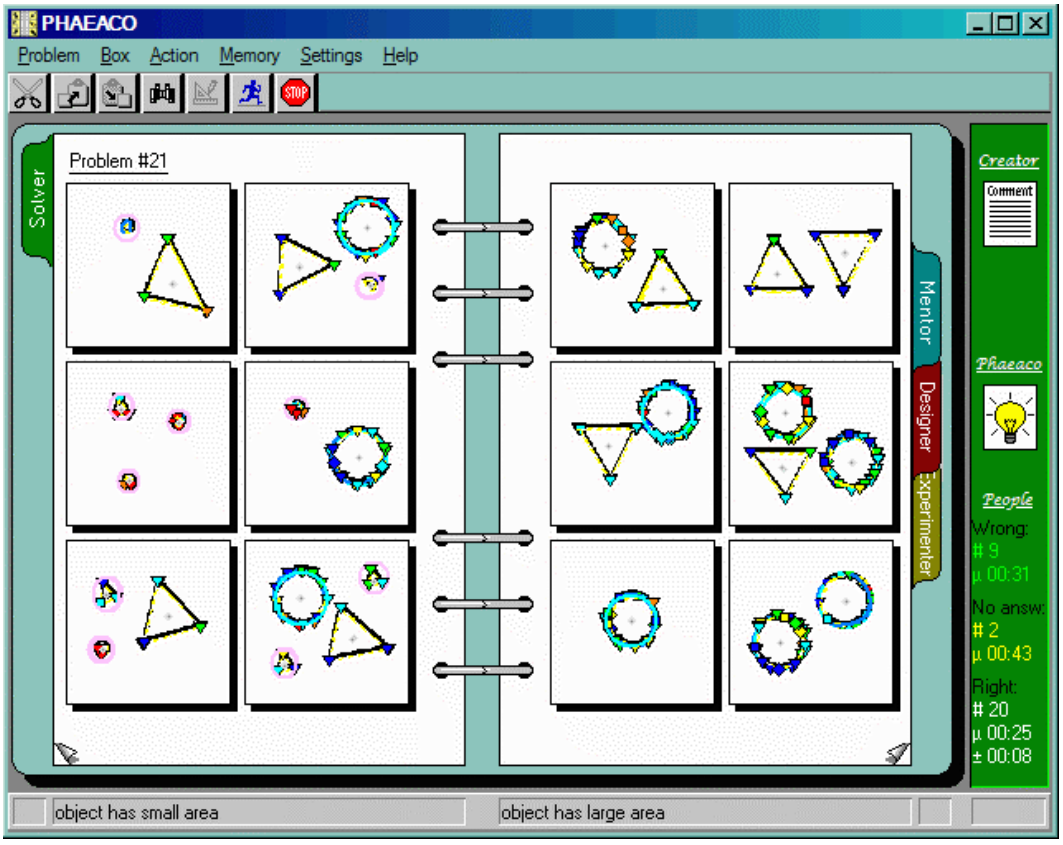


Figure 5.15: BP #21, as is usually solved by Phaeaco

The expected way to express the solution of this problem is: "There is at least one small object in each box on the left, whereas on the right there is never a small object." Phaeaco indeed reaches this solution in its internal representation, but its way of expressing it linguistically is not very sophisticated. Thus, what we read on the left is "object has small area", vs. "object has large area" on the right — a phrasing that can be confused with solutions to problems such as BP #2 (Figure 5.2). Phaeaco notes that there is a contrast in areas among objects within the same box, for some box on the left side (this is true for five out of six boxes). This activates the notions "area", "small", and "large" (among others) in its LTM, which makes it easy to come up with the idea "Let's see if there are small objects only on the left side". (Equally probable is that Phaeaco will examine the idea "Let's see if there are large objects only on the left side", but rejects it as soon as it looks into a box on the right and sees the first large object.) After verifying the idea on both sides, it announces the solution mentioned above. Phaeaco arrives at this solution 34% of the time.

Interestingly, in another 4% of its efforts, Phaeaco arrives at a different solution for BP #21, which Phaeaco's designer was unaware of before the first time it was discovered: it prints the cryptic sentence "every object has a fixed area" on the right side, and nothing on the left (Figure 5.16). This was at first taken to be a spurious wrong answer. But upon closer examination it was understood that Phaeaco acts as follows. It is first attracted by the overall "uniformity" of areas of objects on the right side, in a holistic way. We get this holistic impression if we mentally remove the borders of the six boxes on the right, and see merely the objects spread over the right side of BP #21. At a low, retinal level, it is the *statistical variance* of the population of areas on the right that Phaeaco's routines compute. But at a higher, cognitive level, Phaeaco does

not know about variance, but about the idea that these objects all look similar in terms of their areas; it also notices that the objects on the left do not look similar to each other. This leads Phaeaco to the notion that every object on the right has approximately the same *specific* area. If we were to compute the values of those areas, we would see them ranging between 500 and 600 pixels. Phaeaco does not have access to such numbers at its cognitive level, but it knows the areas are close to some fixed value (whatever that is); and that is what it announces as a solution.

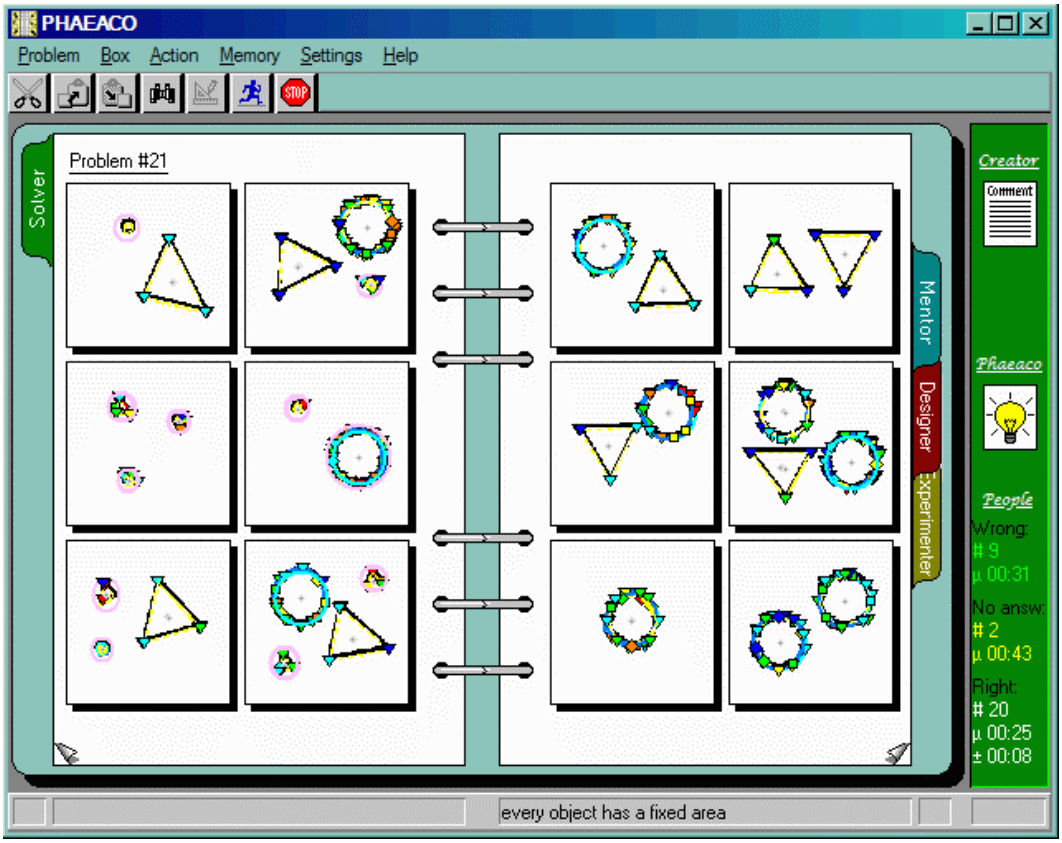


Figure 5.16: BP #21, as occasionally solved by Phaeaco

Human subjects' success rate with BP #21 is different from Phaeaco's: they succeed 65% of the time, give a wrong answer 29% of the time, and no answer 6% of the time.

This example underlines the importance of the separation of levels in Phaeaco's architecture. Numbers such as statistical variance and the specific value of an area are dealt with at the lower, retinal level. The higher, cognitive level benefits from the calculability and existence of such numbers, but has no access to them. If Phaeaco were ever to become a conscious system, it would be able to talk about the rough uniformity of a feature without knowing that at a lower level it has computed the statistical variance of a sample.

5.2 Phaeaco's Mentor

Besides the BP-solving division in Phaeaco's interface, which was presented in all figures in §5.1, there is also the "Mentor division", accessible through the vertical tab labeled "Mentor" (it appears on the right side of the book pages in all previous figures, and on the left side in Figure 5.17). In this part of the interface the person who interacts with Phaeaco (call this person "the mentor" for the purposes of this section) can teach Phaeaco visual patterns beyond those that appear in BP's, and also to associate linguistic descriptions with percepts.

As a demonstration of the usefulness — and the necessity — of including the Mentor division, consider again BP #6 (Figure 1.1). If the LTM lacks the concepts "triangle" and "quadrilateral" (as Phaeaco's LTM in its initial configuration at start-up indeed does), then the only way to approximate the solution to this BP is by noticing the number of sides in each shape (three on the left, four on the right),

or even the number of vertices.[19] But there must be some way to let the system know that, for example, "that kind of figure" is called a "triangle".

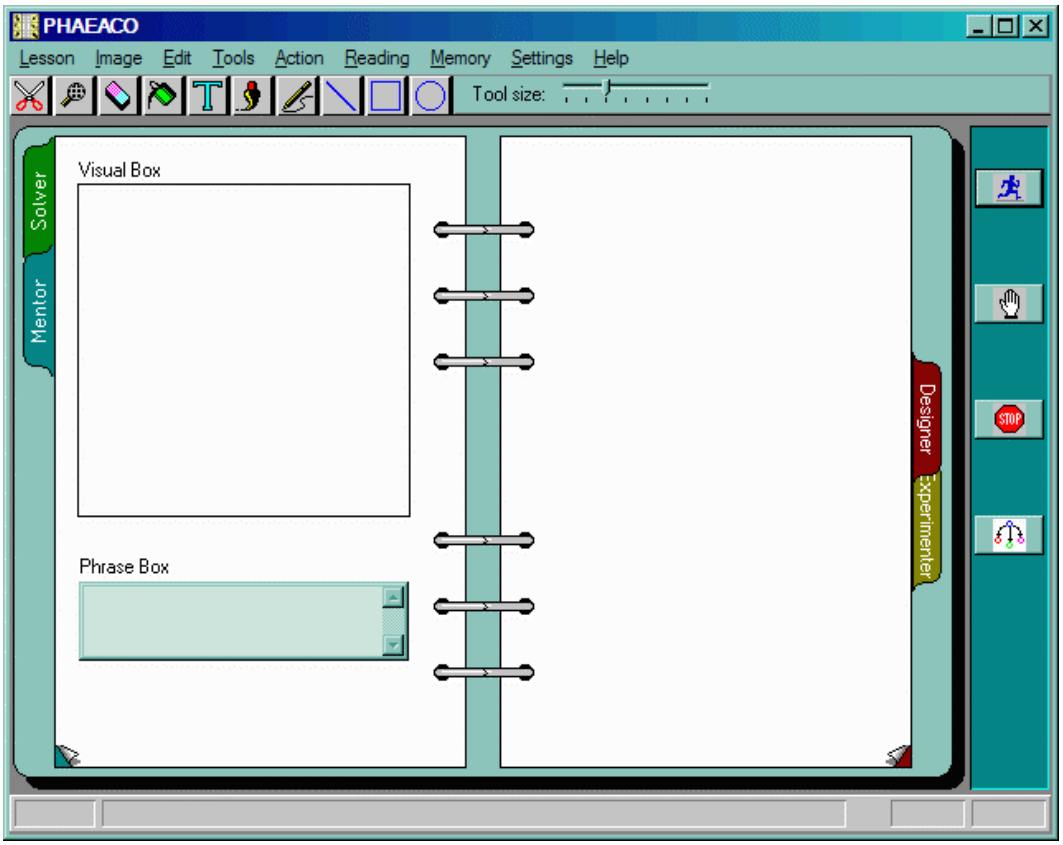


Figure 5.17: The Mentor section of Phaeaco's interface

This is precisely the purpose of the Mentor division. There is a large visual box on the left page, in which the mentor can draw anything at all. Note that although the "tools" of the tool-bar buttons (top of figure) allow only black-and-

[19] Interestingly, of the 26 subjects who solved this problem, only 10 used the concept "triangle". "Three sides" was used 13 times, "three vertices" twice (expressed as "corners" or "points"), and one subject used the combination "3 points, 3 sides" vs. "4 points, 4 sides". Only two subjects used the description "quadrangle" for the right side; of the rest, those who saw triangles on the left, saw either four-sided objects on the right, or "no triangles".

white drawings, it is possible to open and load a true photograph stored in a file, and Phaeaco is able in principle to process that, too, in ways that will be explained in §10.1. Underneath the visual box, there is a "phrase box", in which the mentor can type a phrase, preferably *about* the drawing in the visual box. Figure 5.18 shows an example.

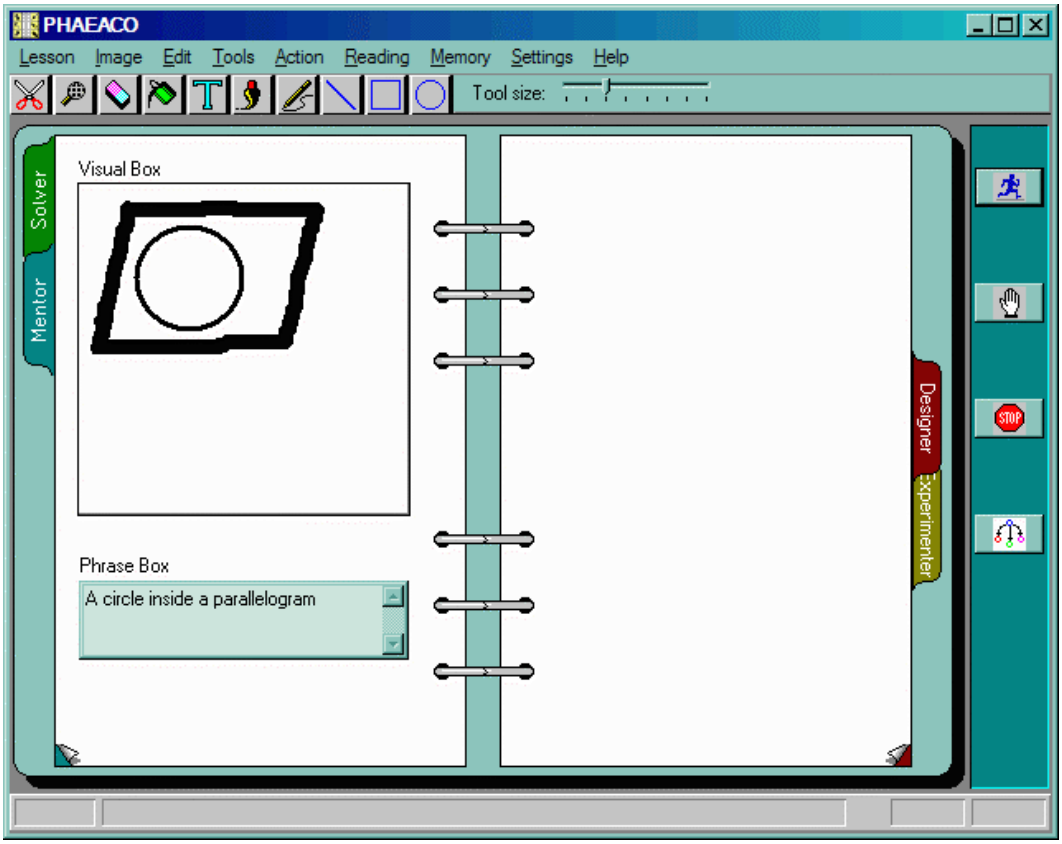


Figure 5.18: A figure drawn and a phrase typed in Mentor's boxes

The phrase can be missing altogether, but if given, it is better that it is related to the contents of the visual box (unless the mentor intends to confuse Phaeaco, or force it to learn associations between percepts and words in some unexpected way). Next, the mentor can ask Phaeaco to process the input (drawing and/or

phrase), by clicking on the button showing a running figure, on the right pane. Phaeaco will take less than a second (in a typical computer) to look at the input, and will end up producing something similar to what is shown in Figure 5.19.

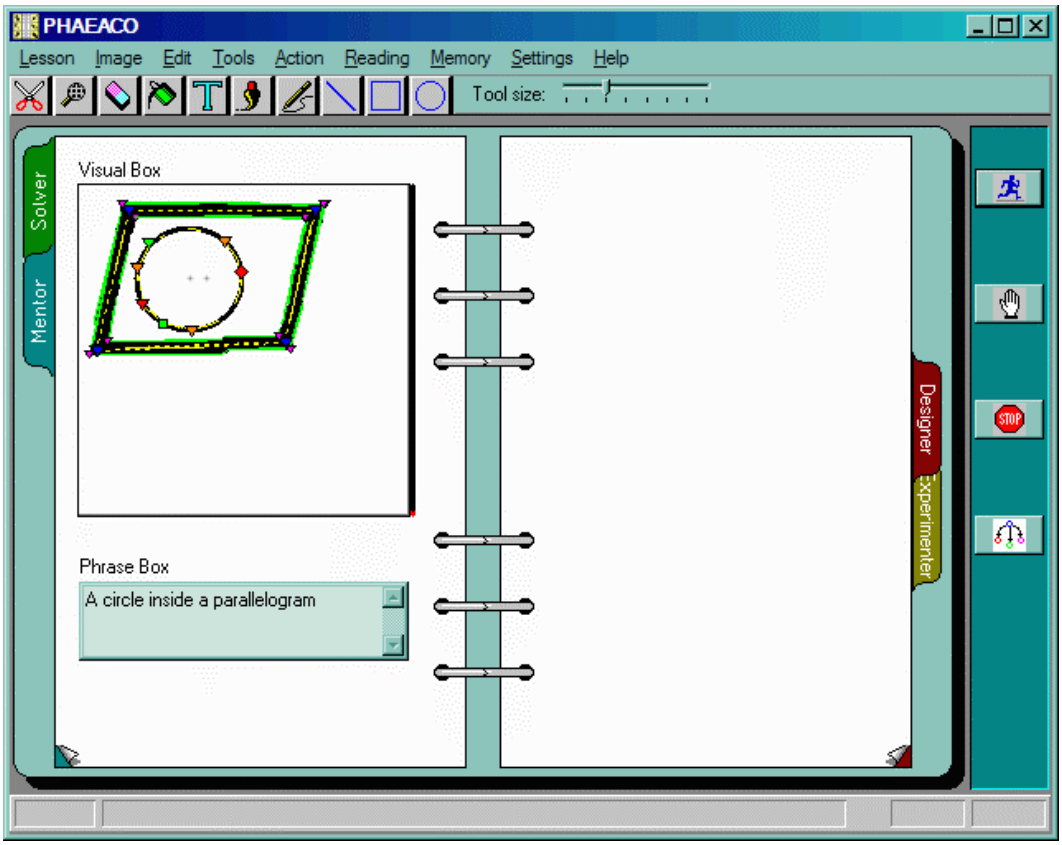


Figure 5.19: Phaeaco's markers of visual processing are superimposed over the drawing

At this stage, Phaeaco has done the following: it has created an internal representation of the drawing, the nature of which will be explained in chapter 7; it has possibly updated or started creating the visual patterns of a circle and a parallelogram in LTM, in ways that will be explained in chapter 8; and if a phrase has been supplied, then Phaeaco has attempted to segment it into morphemes, and made some associations between the percepts of the drawing and the morphemes

of the phrase. The nature of the linguistic processing that must be undertaken by Phaeaco is entirely beyond the scope of the present text, and constitutes a system that rivals in complexity Phaeaco's visual architecture. However, the linguistic system is not complete yet; it is a work in progress that will be described in future publications. At present, suffice it to say that Phaeaco can learn the correct associations between words and percepts after being supplied with several examples that repeatedly make use of the same words (but in different phrases) in the presence of the same percepts. For example, after the previous sample pair of [image, phrase], another pair might show a triangle inside a circle, and the phrase might be: "Now a triangle is inside a circle".

The mentor can add pages to the Mentor division, placing each [image, phrase] pair on a separate page, thus constructing a "lesson", i.e., a sequence of such pairs. Lessons can be saved and re-learned (replayed) at a later time, which allows the mentor to control the order and quantity of concepts learned by Phaeaco before attempting to solve various BP's. (For example, the mentor might want to teach the concepts "triangle" and "quadrilateral" before letting Phaeaco look at BP #6.)

The effectiveness of the algorithm that allows Phaeaco to learn associations between words and percepts has been empirically verified, but its correctness has not yet been formally proven, so its introduction must await future publications.

5.3 Summary

Some of the BP's that Phaeaco can solve were presented, and their properties were discussed, along with other features of the interface of the program. The next chapter introduces the foundations of Phaeaco's architecture.

CHAPTER SIX

Foundational Concepts

The following sections offer an exposition of the foundational background on conceptual representation, paving the way for Phaeaco's architectural organization (to be discussed in next chapter) and explaining its position within this theoretical background. The "classical", "prototype", and "exemplar" theories of concepts are discussed, followed by the Fluid Analogies Research Group (FARG) ideas about how concepts should be represented. FARG principles are directly incorporated in Phaeaco's architecture, whose approach to the representation of concepts can be seen as an amalgam of the prototype and exemplar theories (in spite of their alleged conflict).

6.1 Theories of concepts

6.1.1 The classical theory of concepts

"What are concepts?" is a question that was first asked at least 2,300 years ago. Early answers given by the Greeks are not considered satisfactory today, but they must be mentioned in any discussion of conceptual theories: we must absorb the mistakes of the past to avoid them in the future.

Though Plato is famous for advancing the view of a realm of abstract and pure ideas that exists separately from ordinary perception (the "Theory of Forms"), and for his insistence on definitions in many of his dialogues — most notably in

Sophist (Plato, 1992) — it was Aristotle who first took up the question of what a category, or concept,[20] is. According to Aristotle, a category is what it is because it possesses a set of defining characteristics (Aristotle, 1992). For every category there is an *essence* that makes it "what it is to be" (τό τί ἦν εἶναι). Essences are definitions[21] that are based on a number of elementary categories. Aristotle specified ten such elementary categories, which are akin to *conceptual primitives* in modern theories of conceptual representation, and he gave examples for each. Table 6.1 lists Aristotle's elementary categories (Frede, 1981).

Elementary category (conceptual primitive)	**Description**	**Aristotle's examples**
Substance (οὐσία, τόδε τι, τί ἐστί)	substance "this" what-it-is	human, horse Socrates "Socrates is a human"
Quantity (ποσόν)	how much	four-foot, five-foot
Quality (ποιόν)	what sort	white, literate
Relation (πρός τί)	related to what	double, half, greater
Location (ποῦ)	where	in the Lyceum, at the marketplace
Time (πότε)	when	yesterday, last year
Position (κεῖσθαι)	being situated	lies, sits
Habit (ἔχειν)	habit, possession	is shod, is armed
Action (ποιεῖν)	doing	cuts, burns
Passion (πάσχειν)	undergoing	is cut, is burned

Table 6.1: Aristotle's elementary categories

Moreover, Aristotelian definitions are not concerned with individuals but rather *species* (εἶδος, one of the words Plato uses for "Form") that have essences. A species is defined by giving its *genus* (γένος) and its *differentia* (διαφορά). The genus is a larger set in which the species belongs, and the differentia is what

[20] The words "category" and "concept" will be used interchangeably in this chapter.

[21] Aristotle himself traced the quest for definitions back to Socrates.

distinguishes the species from other members of the same genus. Thus, a "human" might be defined as "an animal (the genus) having the capacity to reason (the differentia)".

Although Platonic and Aristotelian definitions and the theory of essences were held as immutable and authoritative theories of human cognition for well over two millennia, challenges to them began already with their contemporaries.[22] Most notable among those with critical attitudes were the Stoic philosophers, though very few of their writings have survived.

What is more important than philosophical objections, however, is the fact that the classical theory of concepts does not withstand experimental evidence that argues against it. A fundamental claim of the classical theory is that a definition sharply separates those items that belong to the defined concept from those that do not. A study by J. A. Hampton, asking subjects to rate items on whether they belonged to certain categories, showed that, after computing the statistical averages over all subjects, some items were just barely considered category members, while other items were just barely excluded (Hampton, 1979). For example, tomatoes were just barely excluded from "vegetables", while seaweed was just barely included. Similarly, sinks were just barely included as members of the category "kitchen utensil", while sponges were just barely excluded. Seven out of eight categories investigated displayed continuity between purported members and non-members, rather than a sharp division among items.

It might be objected that although statistical averages present a continuum, individual people have a very clear idea of which items belong to which category. But an earlier study (McCloskey and Glucksberg, 1978) showed that people can

[22] Diogenes Laertius (3rd C. AD) recounts one such anecdotal objection: the Cynic philosopher Diogenes of Sinope (4th C. BCE), upon hearing Plato's definition of "human" as a featherless, bipedal animal, plucked a chicken and announced, "Here is Plato's human." (Diogenes, 1992)

change their mind about category membership in ambiguous cases when asked to repeat their judgment after a period of only two weeks.

The objection to the existence of crisp boundaries between members and non-members is closely related to the issue of *typicality* among category members. For example, although dolphins and bats are mammals, they are rarely among the first ones people recall when asked to produce a list of mammals (Barsalou, 1987; Rosch, 1975). Moreover, response times for deciding whether such atypical items belong to a category or not are longer than for the more typical members (Murphy and Brownell, 1985; Rips, Shoben *et al.*, 1973). In addition to being unable to explain the effects of typicality, the classical theory is plagued with other problems. For example, it predicts that the subset relation among categories should be transitive: if horses are a kind of mammal, and mammals are a kind of animal, then horses are a kind of animal. In reality, however, human cognition does not always work in a mathematically describable way. One experiment, for instance, found that subjects regarded chairs as furniture, and car seats as chairs, but usually denied that car seats are furniture (Hampton, 1982).

To address problems such as the ones mentioned above, researchers have proposed ways to mend the classical theory. One such way is to distinguish between *core* and *identification procedures* (Miller and Johnson-Laird, 1976). The core procedures (not to be confused with the term "conceptual core" as used by FARG, to be introduced in §6.2) are still definition-like mechanisms (which remain elusive, since no one has said exactly what they are). In addition, however, there is supplementary information that people store with each concept that serves to identify instances of the concept. Thus, although having fur is not a definitional (core) property of the concept "mammal" (humans have no fur, and there are some furry insects), most mammals have fur, and an animal with fur is quite

likely a mammal. A number of such *characteristic features* aid concept identification, according to this view.

Nonetheless, such remedies to the classical theory are problematic. If only the identification procedures can be examined experimentally, what then is the role of the core ones? What is the purpose of insisting on definitions if an objective way cannot be established to render definitions unassailable, except possibly in contrived[23] cases? After all, changing the classical theory so that it acquires features of its rivals and looks more like them would undermine the reason for keeping it as a possibility.

6.1.2 The prototype theory of concepts

Strong objections to the classical theory and its inability to explain experimental results led to the development of the prototype theory of concepts, largely associated with Eleanor Rosch (Rosch, 1973; Rosch, 1975; Rosch, 1977; Rosch, 1978; Rosch and Mervis, 1975), which was soon followed by the exemplar theory (to be discussed in §6.1.3). According to a review by G. L. Murphy (Murphy, 2002), the prototype theory was at first misinterpreted, taken to mean that the prototype of a category is the "best example" of all members of the category. Thus, the category "bird" would be represented by the best, or the most typical example of a bird. It could be a robin for North Americans, a sparrow for Europeans, etc. This, however, was a deviation from the original proposal, but it persisted, partly because Rosch did not explicitly rule it out in her seminal publications.

---

[23] "Even number" has been proposed as a concept with which people are faster in deciding that 4 is a more typical even number than, say, 7356, but may use the well-known definition to decide correctly with 100% accuracy if given sufficient time (Armstrong, Gleitman *et al.*, 1983).

According to the prototype theory a concept is represented by a set (or list) of features, some of which are assigned greater weights than others. The more often a feature appears in the category and does not appear in other categories the higher its weight is. For example, in the category "bird" the feature "has feathers" will be assigned a very high weight, but the features "has wings" and "has beak" will be given lower weights, because there are other entities that have wings or beaks but are not birds. In addition, features with high variability (e.g., "bird color") will have low weight, because they are only mildly informative. But what is the value of a seemingly discrete but utterly unspecifiable feature such as "bird color"? (What is the color of a parrot of the kind *Ara macao,* a bird that includes all hues of the rainbow on its head, body, and tail feathers?)

Features with a continuous range present an additional problem: how are continuous values to be stored? For example, what is the length of a bird? One possibility is to discretize the range (e.g., small, medium, and large), although this introduces an element of arbitrariness. Another possibility is to store the exact value from a continuous range (e.g., 27 cm), but then how can two values be compared to determine whether they are close enough? This results inevitably in another arbitrary decision. Early proposals, such as storing the average value if the variability of the feature were small (e.g., "length of robin"), but not if it were large (e.g., "length of bird") (Strauss, 1979), were not widely used or accepted.

Another significant problem is the categorization of new items. Specifically, given an item and a category, how do we decide whether the item should be a member? Here there seems to be some consensus, assuming at least that both the item and the category prototype come equipped with a feature list. The algorithm gives points for every common feature among the two lists according to the weight of the feature, and subtracts points for every feature of the item that is not

in the list of the prototype, and for every prototype feature that is not in the feature list of the item (Smith and Osherson, 1984; Tversky, 1977). If the total value thus computed exceeds a fixed threshold (called the *categorization criterion*), then the item is judged to be a member of the category. Problems with this approach include the arbitrariness of the categorization criterion, and what to do with unknown feature values: does a given bird seen frontally have no tail, or is its tail hidden by its body? Is a feature with an unknown value subtracted from the total sum, or simply ignored?

The feature list of a prototype has been criticized as an unstructured, and thus uninformative, data structure (Smith and Osherson, 1984). A bird, for example, is not simply a collection of parts, such as two legs, a head, a body, two eyes, a tail, and a beak, all in a disorganized list. Each part bears a certain relation to other parts: putting the eyes on the body will make a rather poor example of a bird. To rectify this representational weakness, the idea of *schemata* was introduced (Markman, 1999; Rumelhart and Ortony, 1977; Smith and Osherson, 1984). A schema divides up the features of an item into dimensions, also known as "slots", and values on those dimensions ("fillers"). Slots are typed: a slot of type "color", for example, cannot receive a value of type "length". Values can be mutually exclusive: an eagle cannot be both female and male. There can be restrictions on values (up to two eyes is fine for a bird, but three or more is not a possibility), and in some cases values can restrict other values (a bird with wings too short for its body size cannot possibly have the feature "can fly"). Finally, and more important, relations can be explicitly stored in slots: the eyes of a prototypical bird are expected to be *above* the beak, the head is *smaller* than the body, etc.

Now consider how the prototype theory addresses the objections raised against the classical theory of concepts.

- In the prototype theory there is no need for a defining characteristic, since the computed sum total of a number of characteristics with weights is compared against a threshold value. This rectifies the inability of the classical theory to find a single defining characteristic (a definition).
- The observed continuity in membership (the lack of a sharp divide between members and non-members) is explained by the corresponding continuity of the computed sum total that determines membership.
- Borderline membership cases are explained as items that receive approximately equal scores when tested against two different and rival categories (e.g., a tomato as either fruit or vegetable).
- The observation that people often change their minds on borderline cases of category membership can be explained by hypothesizing that people use slightly different weights each time, or a different threshold, or even slightly different features.
- Typical items are identified as members more quickly because they contain the most highly weighted features, so they receive high scores. This rests on the assumption that it is easier to discern a difference between two numbers (when comparing the score against the threshold) when the difference is large than when it is small (§7.3).
- Finally, intransitivity can also be explained, to some extent, by the prototype theory. For example, car seats are kinds of chairs by virtue of one set of features (which includes "can be used for sitting"), but chairs are a kind of furniture by virtue of another set of features (which includes "is located in a room"). Car seats are therefore not a kind of furniture because the two feature lists do not match sufficiently (Tversky, 1977).

6.1.3 The exemplar theory of concepts

Shortly after the prototype theory of concepts was first proposed, an alternative proposal was made, in which the claim was that what is stored in memory is not a summary representation, but a collection of the individual examples that belong to the category (Medin and Schaffer, 1978). This proposal, termed "the exemplar theory of concepts", appeared to be radically different and counterintuitive: the concept "dog", for example, in a person's mind consists of a few hundred dogs the person has seen, some more salient than others.

This immediately raises the issue of how category assignment is made. For example, how do we decide that a flying object that just perched on a branch of a tree in the Eastern United States is a member of the category "goldfinch"? According to the exemplar theory, the input is compared with all concepts stored in memory. The comparison is performed in parallel, hence the result is known practically immediately. The input will thus activate most strongly the stored examples of goldfinches (assuming we already have such examples stored in memory), because those are most similar to the given input. It will also activate, but to a lesser degree, examples of orioles, chickadees, warblers, robins, canaries, and possibly several other similar-looking birds, depending on the observer's prior knowledge. The activation of each of these examples also depends on their salience: a goldfinch is more similar to a canary than it is to a robin, but canaries would be rather low in salience, since they do not normally fly and perch in places where goldfinches do. The input will also activate, probably in decreasing order of strength, examples of sparrows, cardinals, pigeons, chickens, ostriches, chameleons, and so on. Eventually, examples of goldfinches, if most highly activated, will result in accessing the concept "goldfinch" consciously, whereas all other concepts will be primed only subconsciously. Nonetheless, clever

psychological experiments reveal that even such unlikely concepts as "ostrich" and "chameleon" can still receive more activation than other concepts, such as "cathedral" and "democracy", that share no features with "goldfinch".

Evidently, there must be a mechanism for comparing the input to stored examples, resulting in a value of similarity. Medin and Schaffer proposed a multiplicative rule, according to which each pair of compared features results in a value in the range between 0 and 1, but never exactly 0, and these values are multiplied together, resulting in a final similarity value. Each such value is weighted, however, with a weight that brings the value closer to 1 if the feature is insignificant, thus diminishing its effect. For example, in comparing two trees for similarity, it is more important to notice the difference in the shape of their leaves than the difference in their heights, since the feature "leaf shape" is more characteristic of the type of tree than the variable height feature. Thus, the tree-height difference will be multiplied by a weight that will bring it close to 1, effectively neutralizing it.

This gives a procedure for computing the similarity between two items. But how do we compare the input to an entire set of examples in a category? Medin and Schaffer suggested that the similarity scores of the input against each example in memory must be summed up. Thus, if the input is a goldfinch and there are 100 examples of cardinals stored in memory, the 100 similarities of the input to each of the cardinals must be added to yield a final similarity value of the input to the category "cardinal". Even if there are only very few stored examples of goldfinches, their high similarity value will suffice to categorize the input as "goldfinch", because the cardinals, though many, will differ considerably from the input item, and hence their overall similarity will yield a lower value. If several categories turn out to be similar to the input, each category will be

selected with a probability that is proportional to the amount of similarity it has relative to the others (Nosofsky, 1984).

Though counterintuitive, the exemplar theory answers satisfactorily — at least according to its proponents — the issues raised by the prototype theory in questioning the validity of the classical theory. Specifically, the answers to objections are as follows:

- The problem of defining characteristics vanishes in the exemplar theory, since there are no definitions, but only comparisons between examples.
- Typicality is easily explained: the most typical items are the ones that are similar to many category members.
- Borderline cases are explained as items that are almost equally similar to category members and category non-members.
- Typical items are categorized faster than atypical ones: it is easier to find evidence that they are members, since they are similar to a larger number of category members (Lamberts, 1995; Nosofsky and Palmeri, 1997).
- Intransitivity is explained in a way similar to that of the prototype theory: a car seat is similar to many examples of chairs one has seen before; a chair is similar to examples of furniture in many aspects; but car seats are not as similar to examples of furniture in as many aspects.

For all the experimental support it has received, some aspects of the exemplar theory remain unsatisfactory. As the previous discussion suggests, the theory tells us how to proceed once some examples are "stored" and a new input arrives. But how are the examples stored? The prototype theory makes specific, if crude, suggestions: a list of features, a structured schema, etc. Exemplar theorists have left the question of representation blurred and unanswered (Murphy, 2002). What

exactly is stored when, for example, we see a sparrow? Is it the visual image of a specific vantage point taken from a particular angle, a collection of such images, or an entire "motion picture" that includes the sparrow as it turns and pecks on the ground? What if we receive only a fleeting view of the sparrow, or if we see it through the window of a moving car — does it still count as an example, and, if so, how salient an example is it? If we have seen thousands of sparrows in our lives, do they all count as examples? If it makes some sense to claim that the category "dog" is formed by all the specific examples of dogs we have seen (because dog-kinds can be quite different from each other, and we can imagine storing them separately), how much sense does it make to claim that the category "ant" is formed by all the examples of ants we have seen? How can objects as indistinguishable as two ants be stored separately? And how can the mind cope with the amount of information it receives if it stores separately every input object it ever perceives? There have been various attempts at answering these questions, but no overall consensus has yet emerged.

6.1.4 The Generalized Context Model

Over the years, some cognitive psychologists have developed a more sophisticated set of formulas for computing the similarity of two exemplars than the initial multiplicative rule suggested by Medin and Schaffer. These formulas, collectively known as the Generalized Context Model (GCM), are empirical in that they have been tested experimentally and are known to be accurate within statistical error (Kruschke, 1992; Nosofsky, 1992; Nosofsky and Palmeri, 1997). Although the GCM has been employed primarily by exemplar theorists, there is no reason why it should not be employed by alternative representational approaches. Indeed, Phaeaco employs the GCM, explained in detail below. Notwithstanding its grand title, the GCM consists of three simple formulas.

First, the following formula gives the psychological distance d_{ij} between two exemplars i and j that can be compared along features $x_{i1},\ldots, x_{in}$, and $x_{j1},\ldots, x_{jn}$.

$$d_{ij} = \sqrt[r]{\sum_{k=1}^{n} w_k \left| x_{ik} - x_{jk} \right|^r}$$

Equation 6.1: Distance evaluation in GCM

The w_k are weights that depend on the "context"; in other words, these weights determine the importance of each feature x_k. In Phaeaco, each w_k can be in the range from 0 to 1, and r equals 1 (which turns Equation 6.1 into an n-dimensional weighted "Manhattan distance" — see §8.2.1 and §8.2.3).

Next, the similarity s_{ij} between exemplars i and j is given by the formula:

$$s_{ij} = e^{-c \cdot d_{ij}}$$

Equation 6.2: Similarity evaluation in GCM

Thus, the greater the distance d_{ij}, the smaller the similarity s_{ij}. Regarding the parameter c, a high value results in paying attention only to very close similarity, and a low value has the opposite effect. Equation 6.2 is also employed by Phaeaco (§8.2.5).

For the sake of completeness, it should be mentioned that the GCM uses a third formula that calculates the probability P(J | i) that exemplar i will be placed into category J:

$$P(J \mid i) = \sum_{j \in J} s_{ij} \Bigg/ \left(\sum_{K} \sum_{k \in K} s_{ik} \right)$$

Equation 6.3: Membership probability in GCM

K ranges over all possible categories. Phaeaco does not use Equation 6.3, but a different statistical computation, to be explained in chapter 8.

Does the human brain implement the three GCM equations in a literal way, by evaluating summations and exponentials? In principle, such functions are within the capabilities of neuronal processing, as is known in the area of natural computation. It is also possible, however, that the above formulas are emergent results of deeper underlying mechanisms. As an analogy, planets move along nearly perfect elliptic orbits not because they solve differential equations, but because such orbits are emergent properties of gravitational fields: they are 3-D projections of "straight lines" (geodesics) in curved 4-D space-time. Like Kepler in the 17th century, psychologists offer no deeper justification for Equations 6.1–6.3 than agreement with experimental evidence.

6.1.5 Controversy over the correctness of the two theories

Ever since the appearance of the prototype and exemplar theories as heirs to the more or less obsolete classical view, theorists of each of the two new views assumed an antagonistic attitude toward each other. After all, the two theories appear to be totally at odds regarding the explanation for how concepts are formed: the prototype theory supports a summary representation disregarding the specifics, while the exemplar view proposes storing only the specifics. If one theory is right, the other has to be wrong. Countless papers have been published supporting one view, and thereby attacking, at least implicitly, the other. The pattern is that an author performs some experiment, and interprets the results to favor one view. Some time later a new publication appears from the "enemy camp" offering an alternative interpretation of the original results that supports the opposite theory. Consequently, each camp withdraws more deeply into its trenches, and the possibility of a reconciling synthesis becomes ever more remote.

In spite of the ongoing controversy, the approach taken in Phaeaco adopts a middle ground. But how can there be middle ground between two theories so seemingly contradictory of one another? To answer this question, we should note the fact that it is very difficult — if not downright impossible — to distinguish experimentally among the two opposing theories. Consider the analogy suggested by the following figure.

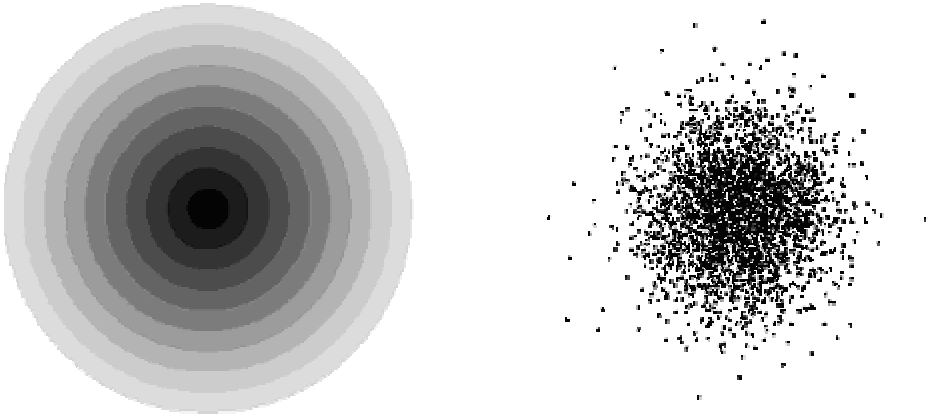


Figure 6.1: Abstraction of the prototype (left) and exemplar (right) representations

The drawing on the left side of Figure 6.1 is an abstract representation of a way the prototype theory could represent categories: through a statistic that includes a mean value and a variance.[24] The drawing is two-dimensional, hence this category contains only two dimensions (features). In general, however, the drawing should be imagined in *n* dimensions. There is a core (dark area) around the mean value, where most members of the category exist, and a halo of progressively[25] lighter regions, corresponding to areas where fewer members exist. In short, this is merely a schematic representation of a Gaussian in two

[24] Note that some prototype theorists consider mean values, but generally not variances.

[25] The shades are supposed to vary smoothly, but the printing of the figure discretizes the range.

dimensions. On the right side of Figure 6.1, the exact same category is depicted, but this time through the individual members (dots) that belong to it. It is not hard to see that most questions that can be answered by the first representation can also be answered by the second, and vice versa. In particular, the mean and variance can be calculated from the individual members in the exemplar case, so the second representation subsumes the first.[26] But the prototype representation can also answer most questions, such as, "What is the probability of finding a member at a given distance from the center?" or, "What is the expected number of members up to a given distance away from the center?" The only questions that are unanswerable by the prototype representation are those that concern individual examples.

As was mentioned at the beginning of this chapter, Phaeaco employs an amalgam of the two paradigms. Phaeaco's representations are primarily prototype-like, with a core and a halo, as will be explained in more detail in chapter 8. But they also include some individual examples, particularly those that appeared earliest in the formation of the category. This means that the first examples of a category are remembered explicitly, at least for a while, during which time they form a statistic (with a mean, a variance, and a few more important parameters). As the statistic becomes more robust, the probability of storing individual examples diminishes. Eventually, no more examples are stored, but the statistic is updated with each new example that is encountered. The following figure gives an abstract depiction of this idea.

---

[26] Naturally so, since the exemplar view is a "lossless" representation in the simplified drawing of Figure 6.1 (although in reality, exemplar theorists do include effects of memory loss due to forgetting). On the other hand, the prototype representation is "compressed".

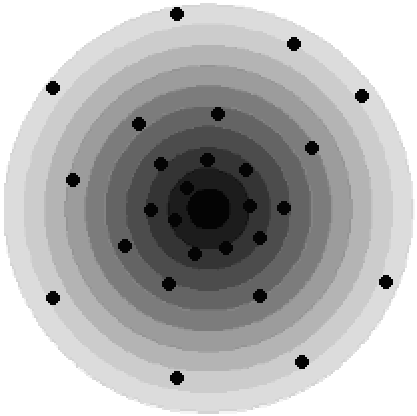


Figure 6.2: Abstraction of Phaeaco's conceptual representation

The justification for this approach is that, although memorization of individual examples cannot be denied, particularly when such examples represent a new concept or idea, Phaeaco has no alternative to compressing its representations sooner or later, because as a non-distributed system it does not have the luxury of employing a scheme in which successive inputs are all maintained independently. In addition, since Phaeaco is implemented and run on processors that are not inherently parallel, it does not have the luxury of running through a large number of explicitly stored cases in order to compute statistical summaries when needed. All this is ultimately a consequence of the difference in the underlying hardware between biological and programmed architectures of cognition, as already discussed in §4.3, and illustrated in Figure 4.9.

6.2 The FARG principles of conceptual representation

At around the same time the first papers appeared proposing the prototype and exemplar views, Douglas R. Hofstadter was working on GEB (1979) (see §1.1). In chapter XIX of GEB, immediately after the introduction of BP's, Hofstadter

drew a figure of what he called “a small portion of a concept network”, redrawn below in Figure 6.3.

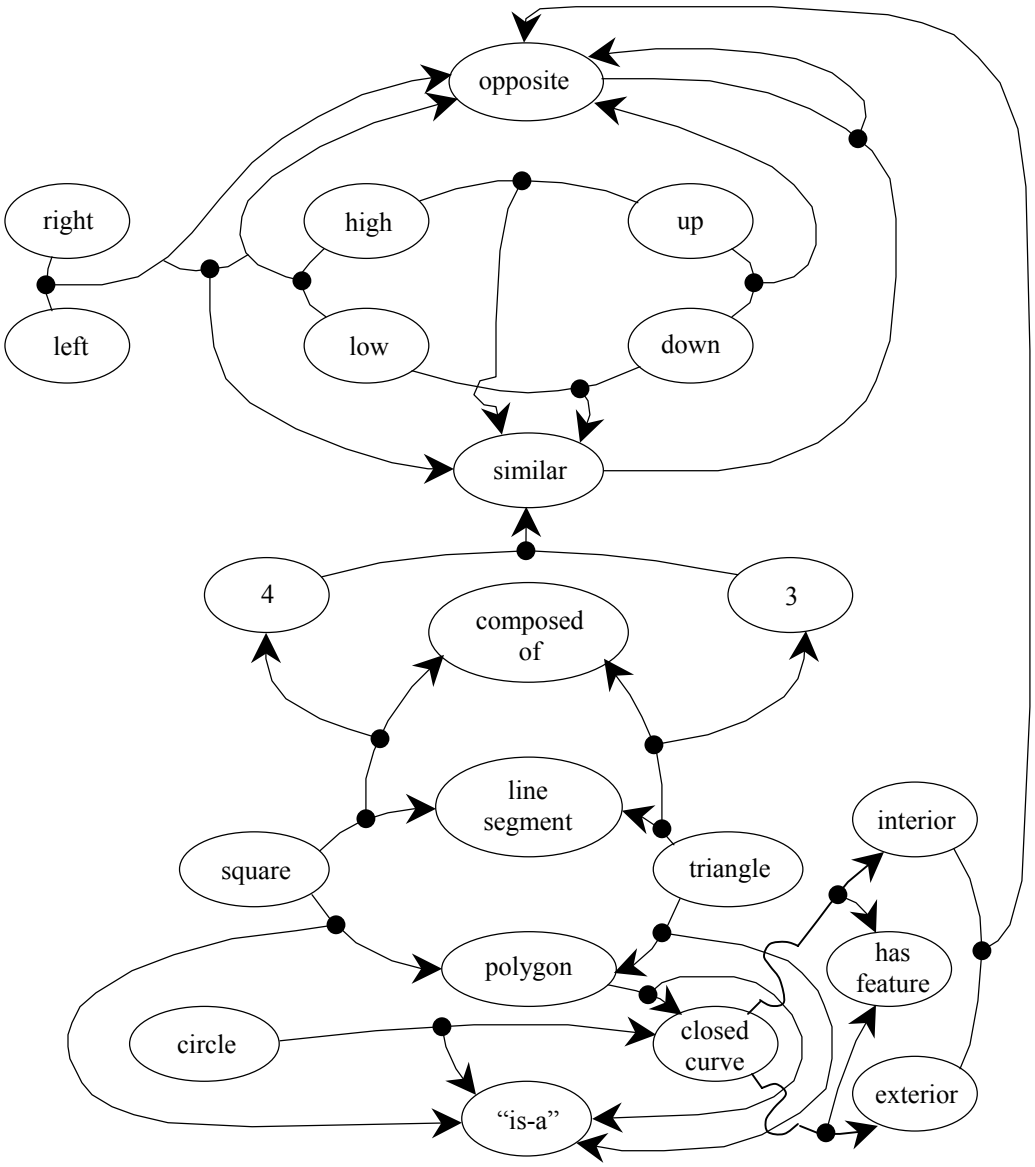


Figure 6.3: A small portion of a concept network, from GEB (Hofstadter, 1979, p. 652)

Each node in the network of Figure 6.3 represents a concept, or, more precisely, the *core* of a concept (a notion that will be explained later). Thus, the node labeled "triangle" represents not a specific triangle that was seen at any particular time, but the core of the abstract ("Platonic") concept "triangle". The concept network is part of the long-term memory of a cognitive system.

Some nodes are linked to other nodes with a directed link. For example, "square" is linked to "line segment" with a link that points to the latter. At some point along the length of this link there is a black dot, from which another directed link starts and points to the node "composed of". Figure 6.4 extracts and shows this portion of the network.

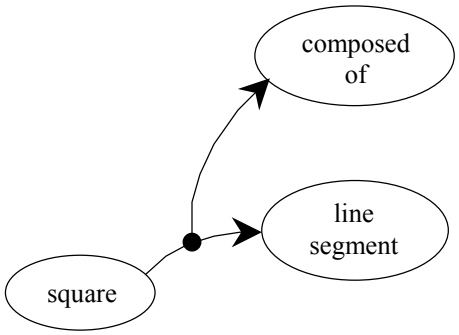


Figure 6.4: An even smaller portion of the network of Figure 6.3

In such cases nodes like "square" can be seen as subjects of a sentence, of which "composed of" is part of the verb, and "line segment" is the object. The entire sentence reads, "a square is composed of line segments". Many such sentences can be inferred from the small network of Figure 6.3: "a circle is a closed curve"; "a square is a polygon"; "a closed curve has interior as a feature"; and so on. (It should be noted that a simple linguistic component that produces such sentences given similar representations is included in Phaeaco.)

Other nodes are linked together through a bi-directional link; for example, nodes "low" and "down", which are "similar", as shown in Figure 6.5. In such cases the concepts are symmetrically related, and the sentence can be constructed in either direction: "low is similar to down", and "down is similar to low"; or even, "low and down are similar".

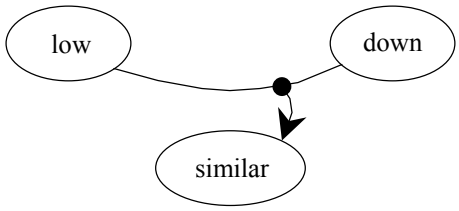


Figure 6.5: A bi-directional link denoting a symmetric relation

Other interesting features of the concept network in Figure 6.3 include some degree of self-reference ("similar and opposite are opposites"), and representation of higher-order relations ("the opposites right–left and high–low are similar").

In GEB, Hofstadter hinted at how these conceptual relations could facilitate the solution of BP's by an automated system. In particular, he introduced the important notion of a *conceptual slippage* between concepts that are sufficiently close in the network. For example, consider BP #24 (Figure 6.6). Suppose that, in solving this problem, at some point we reach the idea that on the right side there are many shapes with line segments. Unfortunately "shapes with line segments" is not enough for reaching the solution, because there are many such shapes on the left side, too. But the concept "line segment" is close to "curve" in the concept network, because the two are "opposites". A momentary look on the left side of BP #24 is enough to let us identify not simply curves, but circles, and the concepts "curve" and "circle" are also close to each other in the network ("a circle is composed of a curve"). Thus, the initial idea of line segments slipped into the

neighboring idea of curves, and then to circles, the existence of which is the solution of BP #24.

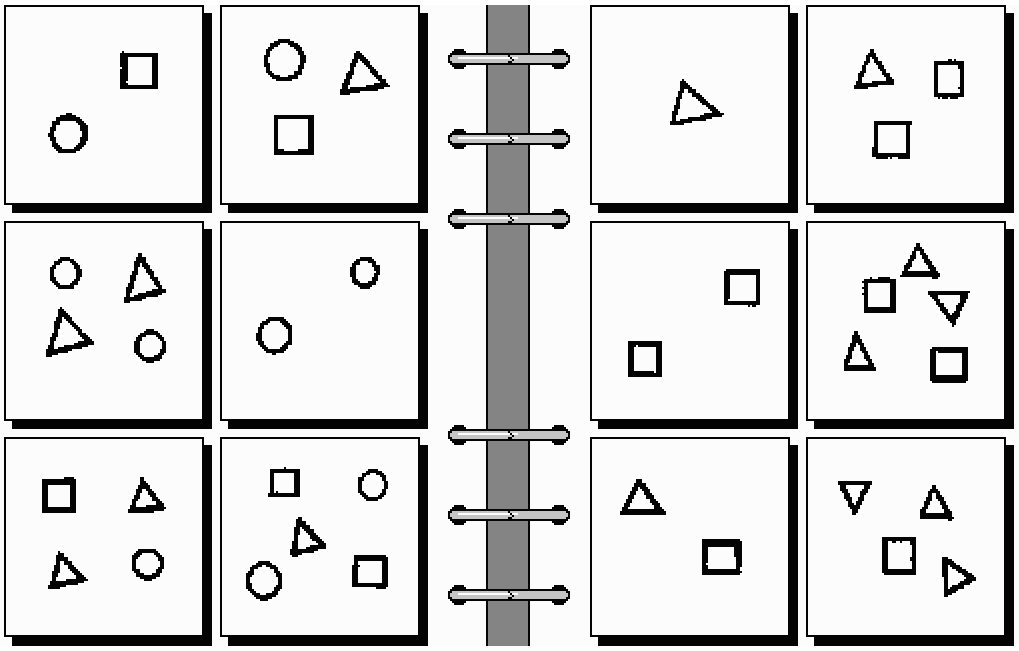


Figure 6.6: BP #24, one of many where a slippage facilitates its solution

These ideas were developed further by Hofstadter and the Fluid Analogies Research Group (FARG), and explained in *Fluid Concepts and Creative Analogies* (henceforth FCCA) (Hofstadter, 1995a). An outline of pp. 211–224 from FCCA follows, since the described architectural principles of the *Copycat* program have largely been adopted in Phaeaco, too.

6.2.1 Copycat and its Slipnet

Copycat was a project conceived by Hofstadter, and as an implemented program it formed the focus of Melanie Mitchell's Ph.D. thesis (Mitchell, 1990; Mitchell, 1993). It was developed further in a second Ph.D. thesis under the name of *Metacat* (Marshall, 1999). The object in Copycat (and Metacat) is to discover

analogies in a letter-string domain. The following is a typical problem of this type.

> Suppose the string ***abc*** were changed to ***abd***; how would the string ***ijk*** be changed in "the same way"?

Though seemingly simple, problems in this domain can be surprisingly subtle. For example, the above problem admits several answers:

- ***ijl***: the most natural answer, replacing ***k*** by its successor ***l,*** just as ***c*** was replaced by its successor ***d***.
- ***ijd***: rigidly replacing the rightmost letter by ***d***.
- ***ijk***: rigidly replacing all ***c***'s by ***d***'s.
- ***abd***: replacing the whole string without regard to its internal structure by ***abd***.

A wealth of other, much more interesting letter-analogy problems led to the development of Copycat's architecture, which was characterized by *fluidity* in the way various ideas about relations among letters and groups of letters were tested, rejected, re-formulated by slipping into related ones, tested again, and so on. One key component of Copycat's architecture was the *Slipnet,* a network of Platonic concepts similar to the one shown in Figure 6.3, but this time employing concepts from the domain of letter-string analogies.

A fundamental idea in FARG's Slipnet is that each node can be *activated* when the corresponding concept in the input is encountered. Activation is a continuous quantity that can spread from a node to neighboring nodes, and in like manner to further ones, but at each jump from one node to another it loses some of its vigor, until — after typically a very small number of jumps — it reaches no

further. Consequently, the *conceptual core* of a concept is defined as the node from which activation starts spreading to neighboring nodes. All affected neighboring nodes belong to the *halo* of the concept. This implies that a concept in Slipnet is not a single node, but an entire — not sharply delineated — collection of nodes, one of which serves as its core.

An additional attribute of the Slipnet makes the behavior of concepts and their activations even more attractive. The links that connect nodes with each other are not passive and featureless "cables" that merely pass activations along. Each link has a label, which is itself a concept in the network (as shown in Figure 6.4, for example). When the label of a link receives activation, the link can be imagined as "shrinking" in a manner determined by the degree of activation value. The notion of "shrinking" is only a visual analogy, and is illustrated in Figure 6.7.

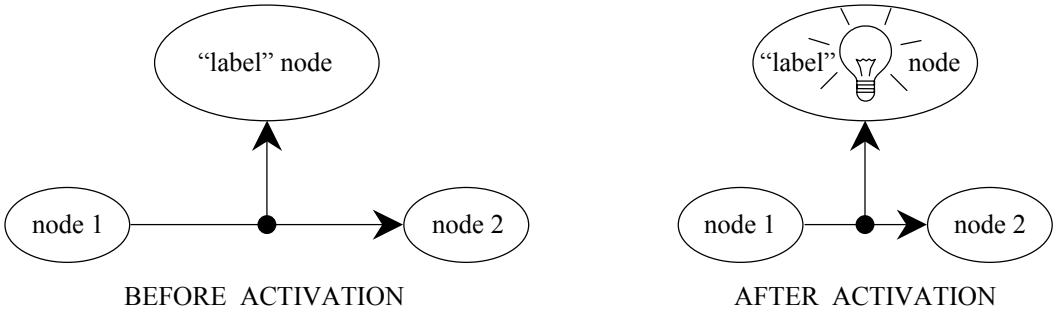


Figure 6.7: Illustration of figurative "shrinking" of a link due to activation of its "label"

As a result of this "shrinking", two nodes that are connected with each other through such a shortened link can be thought of as "coming together", which makes it easier for activation to pass from either one to the other (from "node 1" to "node 2" in Figure 6.7).

For a more concrete example, suppose "interior" is connected with "exterior" by an "opposite" link, as in Figure 6.3. Suppose also that "opposite" receives

some activation. Then the nodes "interior" and "exterior" come closer to each other, and their greater proximity makes it easier to slip from the notion of "interior" to that of "exterior", and vice versa. At the same time, it becomes easier to slip between all sorts of opposite concepts, such as "high" and "low", or "up" and "down", or "similar" and "opposite", etc. Thus, the Slipnet resembles a rubbery structure that shrinks, bends, or is otherwise distorted.

Activations of nodes do not last forever. They fade over time, but the speed with which they fade depends on another parameter of each concept, its *conceptual depth*. The greater the depth of a concept, the more slowly its activation fades. Conceptual depth correlates, but does not coincide, with the abstractness of the concept. For example, "similar" is conceptually deeper than "to the left of", and "polygon" is deeper than "triangle". The reason for the correlation of conceptual depth with the tendency to retain activation is that abstract ideas are remembered best, whereas surface-level, specific memories are easily forgotten.

6.2.2 The Workspace

If the Slipnet corresponds to the long-term memory of human (or animal) cognition, the *Workspace* corresponds to its short-term memory. According to Hofstadter, the Workspace resembles a busy construction site,[27] in which all sorts of structures are continually being built and partially destroyed, to make room for new, possibly larger, stronger, more complex ones. The construction takes place independently at many places, and initially there are only disconnected and very

[27] The visual analogy here is that of "the cytoplasm of a cell, in which enzymes carrying out diverse tasks all throughout the cell's cytoplasm are the construction crews, and the structures built up are all sorts of hierarchically-structured biomolecules." (Hofstadter, 1995a).

simple structures. Over time,[28] however, these simple structures are connected to form larger ones. The initial low-level, bottom-up parallelism is gradually replaced by a higher-level, top-down, focused elaboration of only those structures that appear to be the most salient.

How is an object's *salience* determined? It is a function of two other factors: the object's *importance* and its *unhappiness*. The importance depends on how highly activated the nodes that belong to the structure are; and the unhappiness is greater if the object is poorly integrated with other objects. To illustrate, consider a chessboard with pieces in the middle of a game. Some pieces are more important than others, by virtue of participating in the player's most promising plans. But the player's attention is also diverted from time to time to those pieces that wait "unhappily" away from the action, precisely because they seem abandoned, and the player wonders whether they could be utilized in some other important plans.[29]

6.2.3 Coderack, codelets, and temperature

If structures are built in the Workspace, what is it that builds them? What is the equivalent of a team of workers that expand the edifice by adding parts to it?

The team of workers is implemented by the *Coderack* and its *codelets*. But the worker-analogy is not completely accurate, since the activities of construction workers generally proceed according to a centrally designed plan (the architect's blueprint), and because real workers usually build solely by construction, rather than by demolition. None of these properties is true in a FARG architecture.

---

[28] But note that this time interval, in a cognitive system functioning in real time, might last only fractions of a second.

[29] Interestingly, Alexandre Linhares is currently involved in the design of precisely this kind of chess-playing architecture that employs most of FARG's principles (Linhares, 2005).

Specifically, there is no pre-designed plan that must be followed. If any "plan" can be discerned after some amount of activity, it is an *emergent* one, an epiphenomenon for which no specific prior decision was taken.

The codelets are tiny procedural blocks (they can be thought of as very short pieces of programming code) that wait on the Coderack. The latter is a structure that serves only for accessing and selecting the next codelet to run. Each codelet has an *urgency,* which is a number determining how likely the codelet is to be selected probabilistically from the Coderack: the higher the urgency, the better the chances are for the codelet to be selected. A codelet's urgency is assigned by its creator as a function of the estimated promise of the task the codelet will work on. Once selected, a codelet performs single-mindedly the only task it "knows" how to perform and then it dies. Though codelets are usually implemented in serial computers, thus allowing only a single one of them to act at any moment, their probabilistic selection from the Coderack and their very short life span makes their collective behavior appear as if large numbers of them acted independently and in parallel.

There are bottom-up and top-down codelets. The former act directly on structures in the Workspace without any prior information about what *should* exist there. Top-down codelets, in contrast, carry out actions on Workspace structures with an eye to creating specific types of higher-level structures. Before a codelet dies and is removed from the Coderack, it can manufacture *follow-up* codelets, which, like all codelets, are placed on the Coderack and wait there until selected to run.

In Copycat there is another architectural component that is not implemented in Phaeaco's architecture. It is the *temperature* of the system. The temperature is a measure of disorder in the Workspace. The system starts at high temperature (low

order), at a stage in which hardly any structure has been built, and therefore most codelets are of the bottom-up kind. As time goes by, however, structures are created, and the temperature of the system drops (order increases). The notion of temperature is important in Copycat because its value provides a measure of how much the system "likes" an answer. By its nature, Copycat's domain allows a number of different answers to be given to each problem, with some answers being more "desirable" than others, and much of the system's creativity and similarity to human cognition rests on its sense of which answers are more felicitous than others. In contrast, the BP domain does not lend itself to the same cognitive pressure: BP's usually have a single answer, and if some shapes admit multiple descriptions, Phaeaco manages to reach them without having an explicit measure of their desirability. Naturally, the omission of temperature from Phaeaco's present implementation is not being trumpeted as an advantageous feature; it simply seems that at this stage it is not critically needed. If the architecture of Phaeaco were to be applied to a domain such as letter-string analogies, temperature would have to be added to it.

CHAPTER SEVEN

Workspace Representations

An implementation of the ideas of conceptual representation discussed in the previous chapter is given in the present one. In addition, properties that pertain solely to Phaeaco's representations are explained. The discussion starts with the way the representation of a very simple visual object (a Λ-like shape) forms in the Workspace (§6.2.2) — namely, the formation of a graph-like structure of interconnected nodes. It continues by examining the *activation* that each node possesses, which is an essential element of Phaeaco's representations. One of the representational elements, *numerosity,* is examined in some detail because of its importance both to human cognition in general and to BP's in particular. The rest of the chapter discusses the types of representational elements used in Phaeaco in its current implementation.

7.1 Formation of Workspace representations

The formation of representations of the visual input in the Workspace, which is explained in this section, assumes an LTM that contains only some "hardwired" (preprogrammed) primitives, including such Platonic notions as "point", "line segment", etc. The simplifying assumption that more complex concepts (such as "triangle") are initially absent from the LTM implies that structures are formed in the Workspace with no top-down pressures, built by codelets with no expectations about what should exist in the input. More complex issues of contextual

influences on the formation of representations in the Workspace are discussed in the next chapter.

Suppose the following Λ-like shape is given in a single box of a BP, as shown in Figure 7.1. (Initially, only a very simple figure like this is considered, because the depiction of the representation of any more complex figure would exceed the space available on a single page.) An explanation of the lower-level image processing that occurs in Phaeaco's "retinal level" (a term introduced in §4.3) is postponed until chapter 10. At present, we assume that some initial processing takes place at the pixel level, which in some way gives rise to the higher "cognitive" structures that will be introduced in the present chapter.

Figure 7.1: A simple input figure (from BP #30)

The representations that Phaeaco builds when asked to process the same input at different times are not necessarily identical. But if the input is as simple and decontextualized as the one shown in Figure 7.1, then the representations will always be nearly identical. In general, however, the more complex the input and the contextual pressures, the higher the probability that some variance will exist in the internal representations of multiple views of the same input. This is the case because the processes that build up the representational structure are probabilistic at a very low level (even at the pixel level, as will be explained in the chapter 10). Thus, the points, line segments, and curves that constitute the figure will be perceived in different orders on different runs. Though this different perceptual

order does not result in a dramatic difference in the structural frame[30] of the representation, it might have some effect on the details of the representation, especially in view of the fact that Phaeaco does not allocate an "infinite" amount of time to discover every possible detail that there is in the input. For example, in the input of Figure 7.1, the two lines have approximately equal lengths, the angle they form is nearly 60°, and the overall center of the figure (its "barycenter") is close to the center of the box. All these are details that Phaeaco might or might not include in the representation at different runs. Therefore, the description of the representation of the example in Figure 7.1 that follows cannot be called *the* representation that Phaeaco will construct, but merely *a* possible representation (though a fairly complete one). It is also important to show how the representation is constructed in various stages, rather than simply to display its final form.

The image-processing functions that work at the retinal level are quick to identify pieces of straight lines and to inform the cognitive level of their discoveries. Whereas the numerical details of such discoveries (e.g., lengths of lines, slopes, widths, etc.) remain a property of the retinal level, the cognitive level constructs *nodes,* which are abstractions of corresponding retinal-level details. For example, at the retinal level a straight-line segment can be represented by an equation of the form $y = a \cdot x + b$, where a and b are specific constants, plus a number of other computed values, including the starting and ending point of the line segment. The cognitive level, in contrast, learns simply that "there is a line segment". All the numerical details are inaccessible at the cognitive level, which thus constructs a node representing the knowledge that "there is a line segment". Figure 7.2 shows not only this node, but also one that represents the visual box,

[30] This term is used here informally. The structural frame of the shape in Figure 7.1, for example, can be described as "two line segments that meet at a point". Overall, the representation contains much more information than just the structural frame, as will become evident in what follows.

since every visual input is enclosed in a box in Phaeaco, which thus becomes part of the representation.

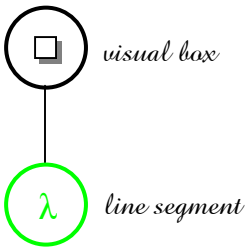


Figure 7.2: An incipient representation: "a line segment in a box"

Nodes will be drawn as circles in such drawings, and they will always contain some depiction of the *type* of the node. For example, the node for the visual box in Figure 7.2 contains a small square, whereas the node for the line segment contains a label with the Greek letter "λ". Such depictions will be used consistently in representational figures, but it must be stressed that they are present in the drawings for our convenience only, to help us identify the type of node. Phaeaco stores internally an integer that corresponds to this type. Each introduced node type will have its description next to it in the figures that follow.

The arc that connects the two nodes is a two-way link. Traced from the box-node to the λ-node, the link must be read as "contains": a box that contains a line segment. Traced in the opposite direction, the link must be read as "is part of": a line segment is part of a box. Thus, links have types, too, but the labels of their types ("contains", "is part of", etc.) will be suppressed from our drawings in order to avoid cluttering them. Some additional properties of links will be explained later. For now we note that if the link is one-way only, this will be signified with an arrowhead at the destination node. The lack of an arrowhead implies a two-way link.

Shortly after constructing the first link, the retinal level discovers not only the second line segment, but also the fact that the two line segments both belong to a single connected component, which will here be called an *object*.[31] Informed about this, the cognitive level creates a node that denotes precisely the existence of an object that contains the two line segments. But in order to do so, it must reconfigure the structure of Figure 7.2, because it is now known that the previously identified line segment is not simply a part of the box, but part of an object, which in turn is part of the box. Thus, an object-node must be inserted between the box-node and the λ-node. Figure 7.3 shows not only the object-node inserted, but also the second line segment as part of the same object.

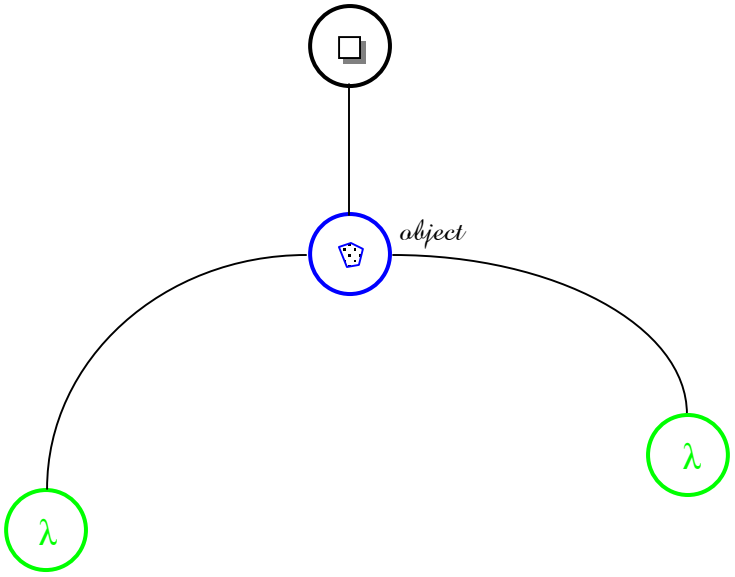


Figure 7.3: A box with an object that contains two line segments

Objects will be represented in the drawings that follow by nodes with a dotted irregular shape, as in Figure 7.3.

[31] Isolated line segments have their λ-nodes connected directly to the box-node.

Up to this point, all nodes at the cognitive level have been created because the retinal level "said to do so". In other words, there was strictly bottom-up processing of information, with the retinal level dictating what must be constructed at the cognitive level. The construction of each cognitive node, however, entails the creation of a number of codelets (§6.2.3) in the Workspace, each specific to the type of the introduced node. For example, each λ-node will add to the coderack six codelets, "wanting" to represent the slope of the line segment, and also its length, width, extremities (endpoints), and midpoint, as well as a count of how many line segments there are in this object (or whatever larger component contains this line segment, in general). Naturally, the urgencies of such codelets vary widely. Slope and length codelets, which have very high urgencies, are selected from the coderack almost immediately.[32] The line-counting codelet has a medium urgency, while the width, extremity, and midpoint codelets have very low urgencies, so they only have a decent chance to be selected and to work if there are contextual pressures, i.e., if concepts such as "line width", "middle", or "line end" have been primed in LTM. In the absence of such contextual pressures, it is almost certain that the activity in the Workspace will come to an end before these codelets are ever selected. (Soon it will be explained what causes the activity in the Workspace to cease.)

Another important issue, before the activity of some codelets is exhibited, concerns the origin of codelets. Who decides how many codelets a line segment gives rise to, and of what type? Why does a line segment create exactly the six types of codelets listed above? If this is a predetermined, "hardwired" decision,[33]

---

[32] An architectural decision reminiscent of (and inspired by) neurons acting as slope and length detectors in area V1 of the visual cortex of the brain.

[33] A "hardwired" decision is one that is determined rigidly by specific lines of programming code.

what if in the future a new (hence, overlooked) feature of the concept "line segment" were discovered, necessitating a new type of codelet to work on it?

The answer is that in Phaeaco the decision of which codelets to build, given a representational node (e.g., a λ-node), is not simply hardwired. For example, the particular Platonic node that stands for the concept "line segment" in LTM is linked to a number of other concepts, such as "slope", "length", "width", "extremity", "middle", and "numerosity". Given a λ-node, Phaeaco goes to the Platonic node "line segment" in LTM and creates codelets corresponding to all Platonic nodes ("slope", "length", etc.) that are linked to "line segment" according to a particular type of link. These linkages are of course pre-manufactured, so in a sense they are hardwired. But nothing prevents the system from creating a new linkage in the future after learning something new about line segments. Phaeaco's LTM is both hardwired (non-empty at startup) and expandable (new concepts and linkages can be added as the system processes information). Thus the answer to the question of the rigidity of codelet types is that Phaeaco's ability to "learn" (i.e., expand and modify its LTM) is responsible for adding future behaviors that were not inherent in its initial programming code.

This pertains to a deeper issue that deserves a brief additional comment. A common misconception among lay people, and even among computer scientists who do not work in AI or cognitive science, is that "computers can do only what they have been programmed to do". By implication, computers will remain eternally dumb, unable to demonstrate the flexibility and creativity of intelligent human minds. Indeed, for programs that include only their lines of programming code, this is trivially true. But a program such as Phaeaco includes its memory as well, which is permanently modifiable by what Phaeaco "sees". In principle, what Phaeaco "sees" cannot be predetermined, since it might process pixels that arrive

through a camera focused on the external world. Thus, Phaeaco's LTM, and by extension its behavior, abstractly reflects the history of its transactions with its environment. The unpredictability of the latter implies that Phaeaco's behavior cannot be considered entirely preprogrammed.

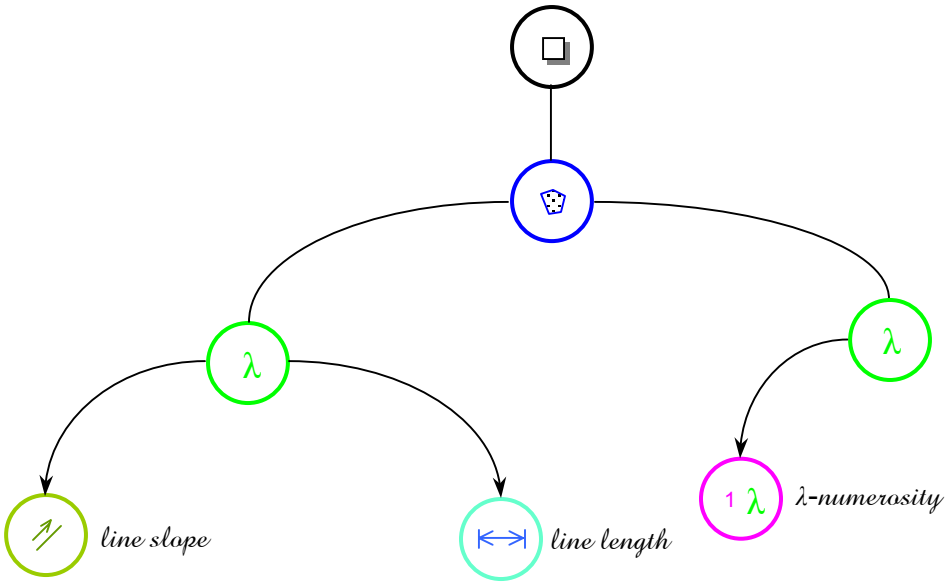


Figure 7.4: The representation as it looks after the work of some codelets

Figure 7.4 shows a further stage of the modification of the representation of the input, after a small number of codelets (all elicited by the λ-nodes) were selected from the coderack and run. The λ-node on the left is connected to two additional nodes, one representing the slope of the line (left) and one representing the length of the line segment (right). The λ-node on the right is connected to a single additional node, representing the fact that the program has "seen" and counted only one of the two lines, so far.[34] Note that the connecting links are now unidirectional, shown in the diagram by arrows ending on the linked nodes. There

[34] Not necessarily all entities need be counted, especially if they are many, as will be explained in §10.3.15. Also, if *m* things are later seen as *n,* a codelet will be launched to update the count.

is no particular reason for selecting these features (slope, length, and a λ-counter) to show in Figure 7.4; a more or less random sample of them was selected, just as the system selects and runs the corresponding codelets randomly.

The nodes for slope and length, shown in Figure 7.4, are of a type called *feature nodes*. The characteristic of such nodes is that the programming structure that implements them includes not simply a number (for the angle of the slope, or for the length), but a small set of statistics, defined as in Table 7.1:

Field	Description
N	number of observations
mean	average value of sample data
var	variance of sample data
sum_sqr	sum of squares of sample data
min	minimum value of sample data
max	maximum value of sample data

Table 7.1: Programming structure for the statistics of a feature

The reason for maintaining a set of statistics as opposed to a single number for a feature node is not apparent at this early stage in the representational buildup. It looks as if a single observation made by a single codelet should result in a single number. Although this is true, Phaeaco does not make single observations but repeated ones, even on the same input (the details of this mechanism will be explained in §11.1.3). Thus the structure of a feature node must accommodate a set of statistics, instead of just one single number. The advantage of employing statistics will be explained in the context of pattern formation (§8.3).

Slopes and lengths are examples of continuously varying features. There are also discrete-valued features, to be discussed soon.

The λ-node on the right of Figure 7.4 is connected to a different type of node, called a *numerosity node* (marked by the symbol "1 λ" in the figure). The concept of numerosity (§1.2.1, §5.1.1) is a familiar one in psychology and cognitive science, and a large number of books and dissertations have been devoted to it. A brief overview of numerosity in general, as well as a detailed description of how Phaeaco deals with it, is given in a later section (§7.3). Here it suffices to say that although a numerosity node is not the same as a feature node, it shares with the latter the characteristic of being described by a set of statistics (Table 7.1), rather than by a single value.

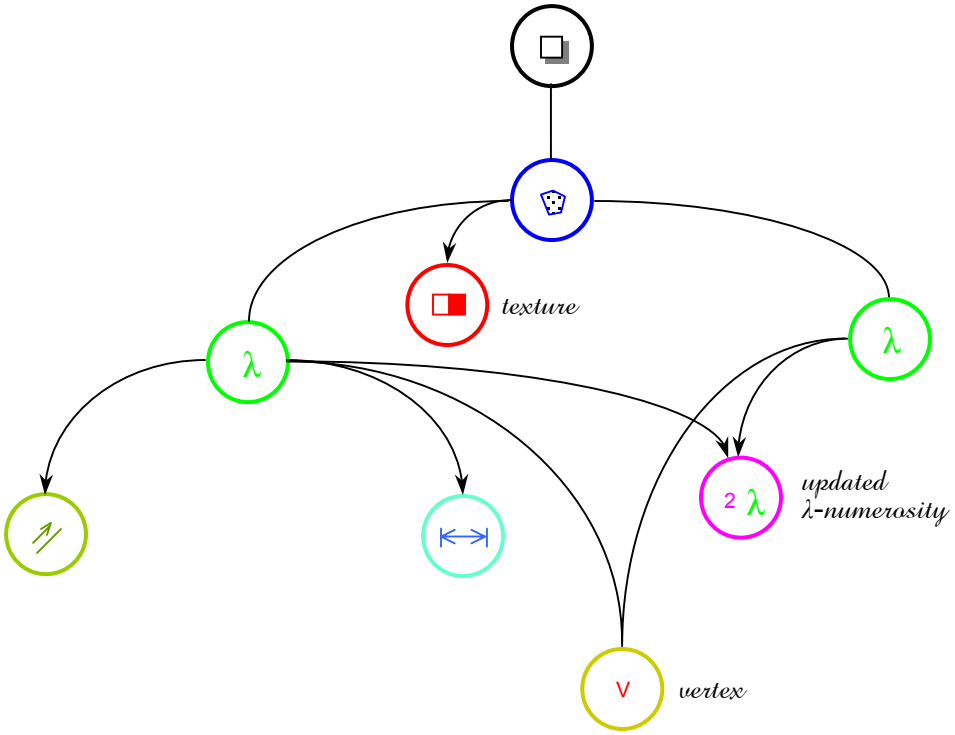


Figure 7.5: Further enrichment of representation: texture, vertex, and updated λ-numerosity

In Figure 7.5, more nodes and connections have been added to the representation. First, there is a *texture node,* linked directly from the object node. This is one of the feature nodes, and was added because a codelet ran that "wanted" to examine the "texture" of the object. (This codelet was put on the coderack by the codelet that created the object node, but only now was it selected to run.) A texture is a discrete-valued feature in the domain of BP's. Objects in BP's are usually either *outlined* or *filled*. These are the only two values that a node of this type can represent. Naturally, the texture of an object can be a much richer notion than simply "outlined" or "filled", as BP's #196 (Figure 2.12), #97 (Figure 2.7), and #180 (Figure 1.15) clearly demonstrate. In such complex cases the texture cannot be represented by a simple value but requires an entire representational sub-tree that would be described, for example, as "parallel lines slanted at 45°, narrowly spaced" (see box I-D of BP #97, Figure 2.7, for an example). The texture nodes presented here make no attempt to encompass such complex notions.

The texture of the Λ-shaped object examined here is perceived as "outlined". But a question immediately arises: How thick can a line be before it is perceived as possessing an identifiable shape, and therefore no longer as a line but as a filled object of that shape? Figure 7.6, below, illustrates this point.

Figure 7.6: Line segments or filled objects?

Assuming there is no contextual pressure, Phaeaco's retinal level has a probabilistic threshold of "thickness" (concentration of black pixels) beyond which it perceives a thick line as a filled object. This threshold has been manually

fine-tuned so that it corresponds approximately to the threshold beyond which the human eye also tends to see a filled object, rather than a line, from a reasonable distance.[35] Under contextual pressure, however, the value of this threshold can be pushed up or down, which is an example of how higher-level contextual pressures can cause a change in the functioning of the lower-level retinal procedures.

There is another issue concerning the statistics of discrete features. All feature nodes, as mentioned earlier, have a set of statistics as given in Table 7.1. But if a feature can take on discrete values only, what sense does it make to compute a set of statistics? Is it necessary to store an average, a variance, etc., for a binary feature such as texture? The answer is that when a single object is considered, naturally the value of its texture is either "outlined" or "filled". But when a *pattern* is formed, as will be explained in §8.3, several objects of different textures might be matched together to form an object of "average" texture. Internally, Phaeaco assigns the value 0 to "outlined", and 1 to "filled", so an average texture might have the value 0.75. This is not devoid of meaning; it is as meaningful as the statement that the average family of a given population has 2.15 children. These ideas will become clearer in chapter 8. For now we simply note that statistics on discrete features are useful in general.

A second addition to the representation in Figure 7.5 is the *vertex node,* which is shown at the bottom of the figure and has the label "v". Note that both λ-nodes are linked to it with a two-way connection. Such nodes are *not* the result of codelet activity, but of further processing at the retinal level.

---

[35] The total length of a thick line might also play a role in whether the human eye perceives a line or a filled object, as the rightmost examples of Figure 7.6 suggest. For small objects, Phaeaco uses a magnification algorithm that attempts to replicate a close-up view. If the object is too tiny, Phaeaco cannot compete with the human eye, and will perceive a "dot" instead, i.e., a point with a certain size (more about dots and points in §10.3.10).

Finally, a third modification of the representation in Figure 7.5 is the updating of the λ-numerosity node, which now is linked with both λ-nodes and has the label "2 λ". The new connection was created when the codelet that was assigned the task of counting the leftmost λ-node was selected and run. The codelet located the already existing λ-numerosity node in the structure (labeled "1 λ" in Figure 7.4), updated it, and linked it with the λ-node.

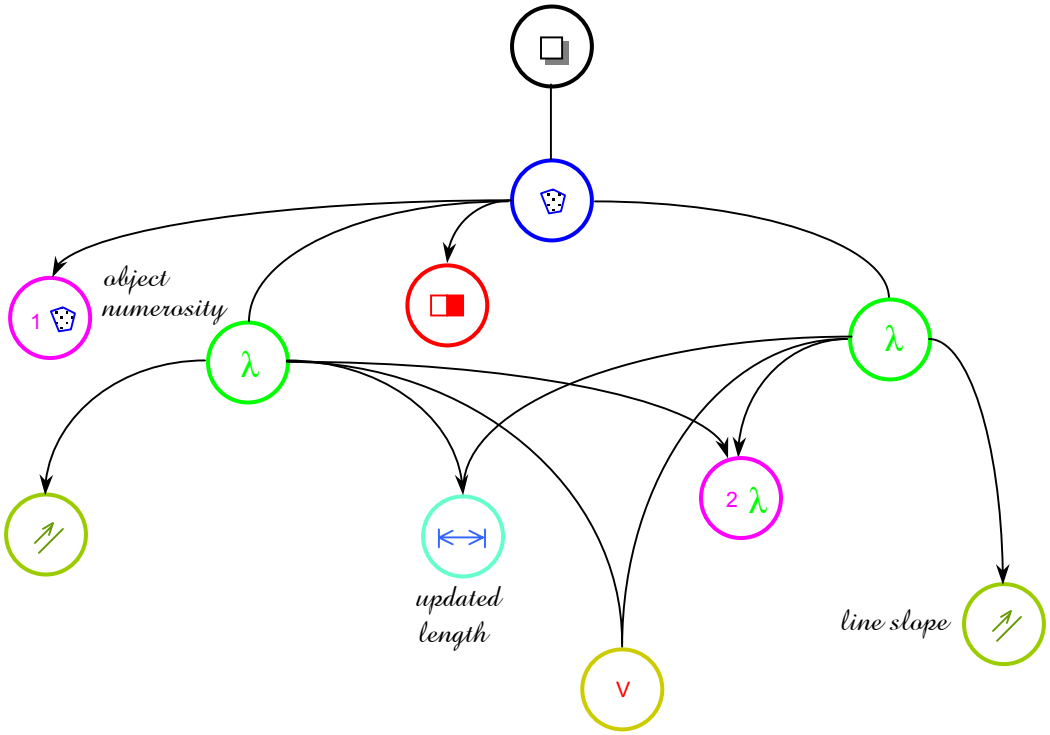


Figure 7.7: A new line slope, object numerosity, and updated line length

Some additional nodes are shown in Figure 7.7, all the result of further codelet activity. The object has been counted, building the *object numerosity node* into the structure. Thus, each numerosity node "knows" (stores internally) not only the quantity involved but also the type of the counted entity. A second modification is that the slope of the rightmost λ-node has been noticed. Finally, the node that

represents the line-segment length is now linked with both λ-nodes. This is because the two line segments have approximately the same length. The word "approximately" here means the following. At the retinal level, two quite accurate numbers have been computed as lengths of the two line segments. At the cognitive level, however, there is a threshold of similarity for lengths beyond which two lengths are treated as exactly equal. This threshold has been manually set so that, in the absence of contextual pressures, it treats as equal those lengths that the human eye would probably not distinguish. As in the case of line widths, however, the context might modify the value of this threshold. For example, if there is sufficiently great pressure to see a triangle as isosceles, but its two sides (other than its base) are not exactly equal, lowering the threshold of acceptance of length equality will allow the two nearly equal sides to be treated as equal.

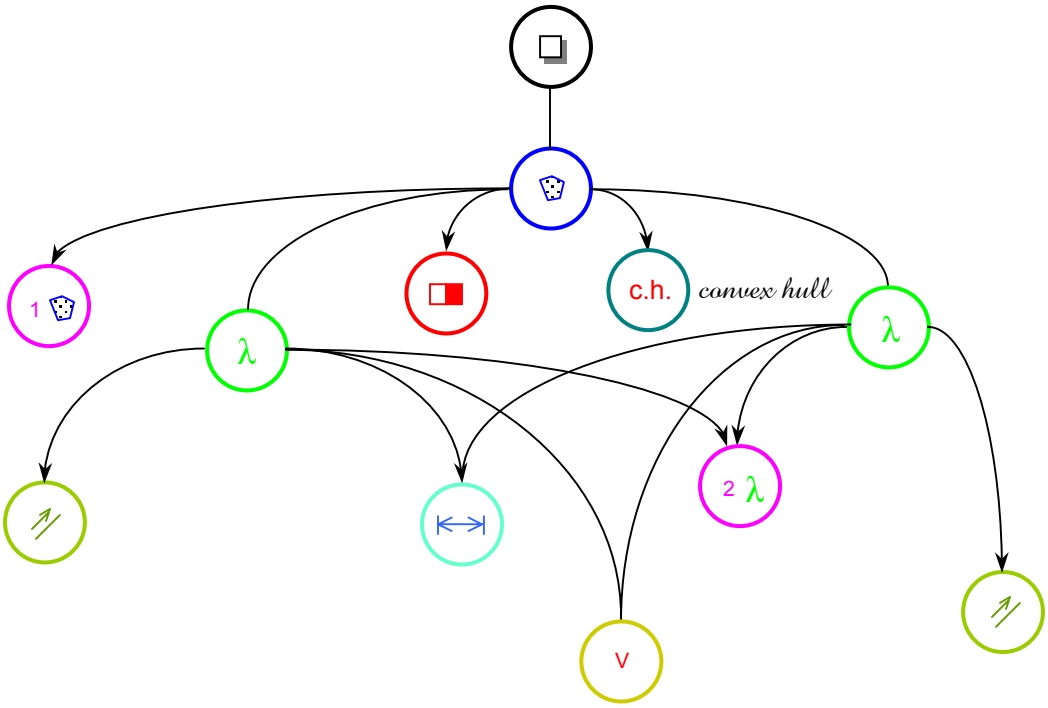


Figure 7.8: Convex hull added

Figure 7.8 shows a single added node representing the *convex hull* of the shape. The convex hull of a set of points can be defined geometrically in more than one way. For example,

- it can be the smallest closed figure that includes all points, such that the tangent to any point on its perimeter does not intersect the figure; or,
- it can be the smallest closed figure that includes all points, such that if we choose any point in its interior and draw a straight line in any direction, the line will intersect the perimeter of the figure at exactly one point.

Here we do not need to use either of the above definitions, however, nor are they necessary for Phaeaco. An algorithm for constructing the convex hull of a set of points that starts with three points and proceeds incrementally with additional points is given in §10.3.18. An easy way to visualize the convex hull is by imagining the shape that a tight rubber band would take if stretched so as to include all points.

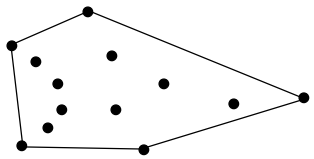


Figure 7.9: Convex hull of a set of points

How immediate is the perception of the convex hull in our cognition? Do people immediately recognize it and perform cognitive manipulations on it, or is it something seldom noticed? As the experimental data in Appendix A show, the answer is probably somewhere in the middle. Of the 31 subjects asked to solve BP #4, which involves the notion of "existence of depressions", 17 did not answer

at all, nine gave a wrong answer, and only five (or about 16%) answered correctly. One might argue that using the idea "existence of depressions" is not the same as "seeing" the convex hull itself. There is also BP #12 (Figure 7.10), however, in which the perception of convex hulls is probably involved more directly. On the left side, figures are clearly elongated; on the right side, most figures are not elongated, except one, in box II-C. That figure can be seen as "not elongated" only if the convex hull around it is perceived. To reach the solution, one must therefore perceive the convex hull around all other figures. Of the 30 subjects asked, 21 supplied no answer, two gave a wrong answer, and seven (or about 23%) answered correctly.

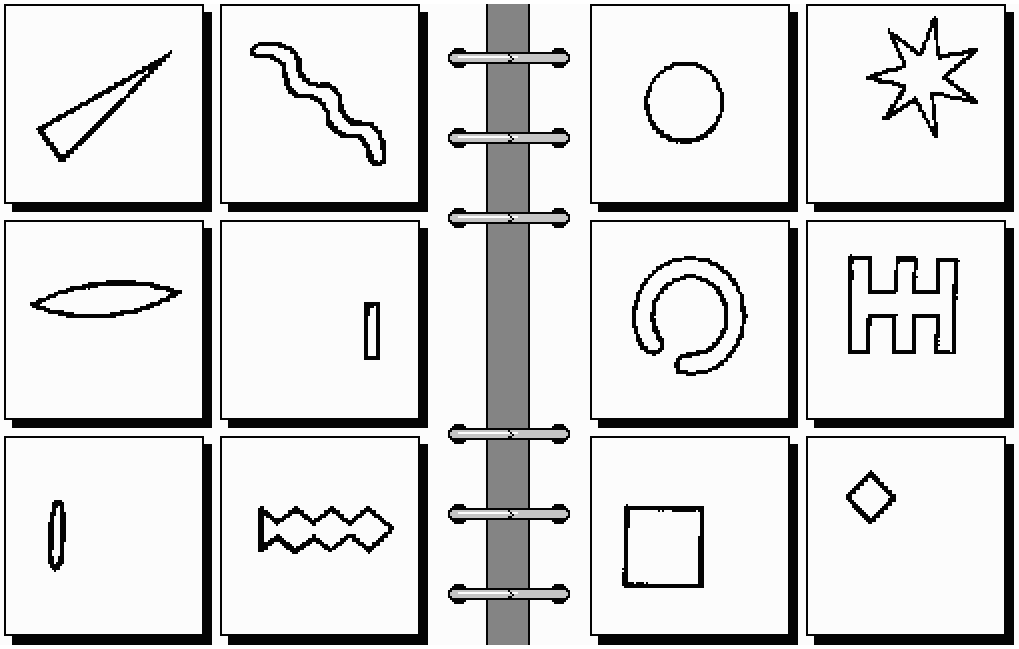


Figure 7.10: The figure in box II-C of BP #12 is not elongated only by perceiving its convex hull

These results suggest that the convex hull is a percept that people are capable of perceiving, albeit not easily. Accordingly, Phaeaco is not very eager to "see" it,

and chronologically it is one of the last perceptual elements that are built into a representation. This is because of the low urgency of the convex-hull perceiving codelet, which is put on the coderack along with other codelets that result from an object node. Nonetheless, once the convex hull is perceived and its node is added to the representation, it results in the addition of a few more codelets into the coderack. One of these codelets with relatively high urgency computes the area of the convex hull, depicted in Figure 7.11, along with one more element.

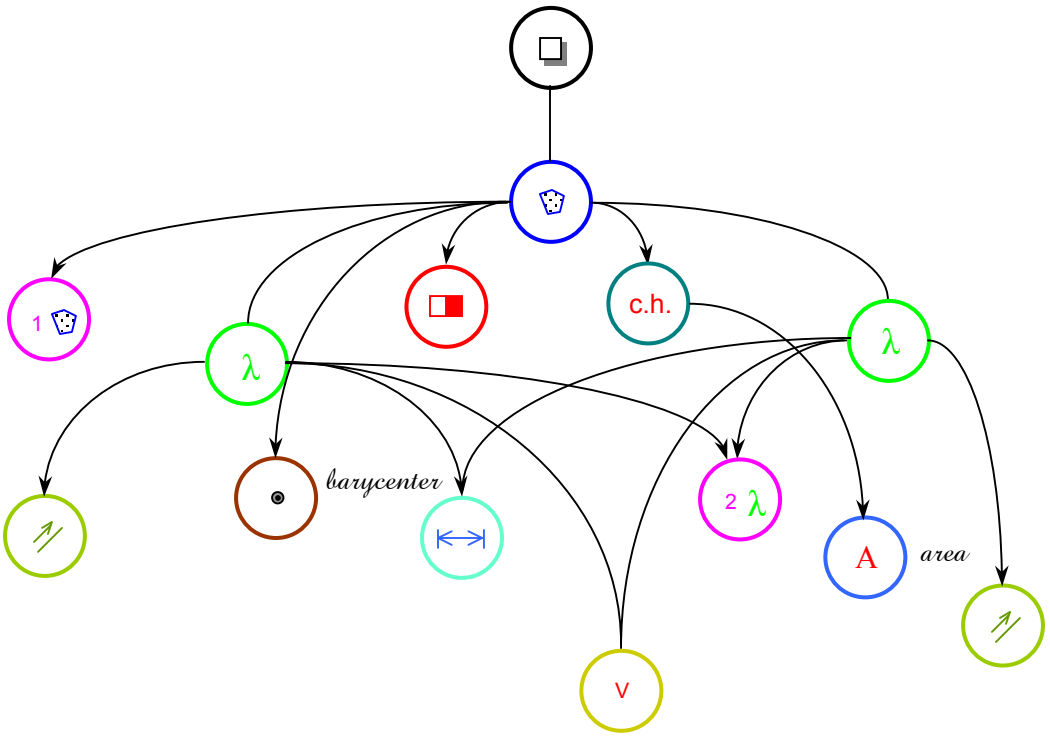


Figure 7.11: Area of convex hull, and barycenter of object

It should be noted that with the introduction of the notion of convex hull we have entered the domain of *imaginary* percepts. The convex hull as an "object" is not real, but imagined mentally. Equally imaginary is its area, which tells us how "large" the Λ-shaped object is, although it consists of only two line segments.

Figure 7.11 shows one more imaginary percept: the *barycenter,* or center of gravity of the object. This is the point on the plane where the object would balance if it were a physical object and each of its pixels had the same mass. This point seems to be relatively easy to perceive, as the data on BP #8 suggest.

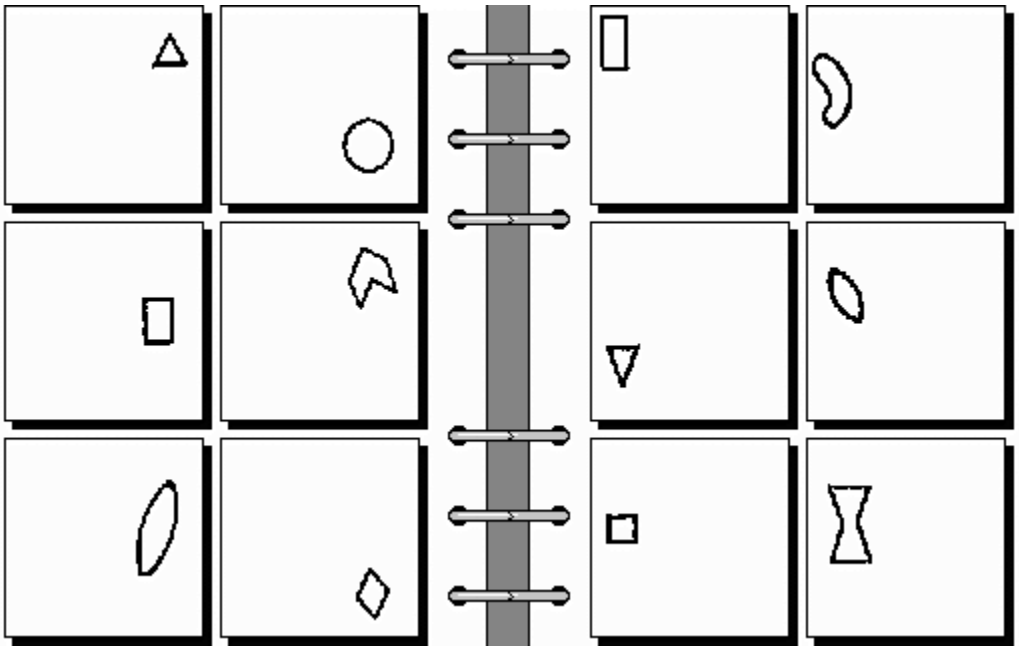


Figure 7.12: BP #8, solved by noticing the placement of barycenters within the boxes

BP #8 (Figure 7.12) was solved by 24 out of 31 subjects (around 77%), whereas the other seven subjects did not provide an answer (see Appendix A).

If barycenters are somewhat easy to see in isolation, perceiving them becomes compelling if the objects are lined up in some fashion, as BP #84 indicates (Figure 7.13). This problem was solved by 100% of the subjects (31 out of 31), and in a relatively short time (average 13 sec). The lined-up objects in BP #84 are very small circles, and we group them together (separating them from the somewhat larger square in each box) because they are so similar (more on this

mechanism in chapter 8). When a group with a number of objects is perceived, one of the things that Phaeaco does is to abstract the objects with their barycenter, which leads it to perceive the shape formed by their barycenters (§10.3.15).

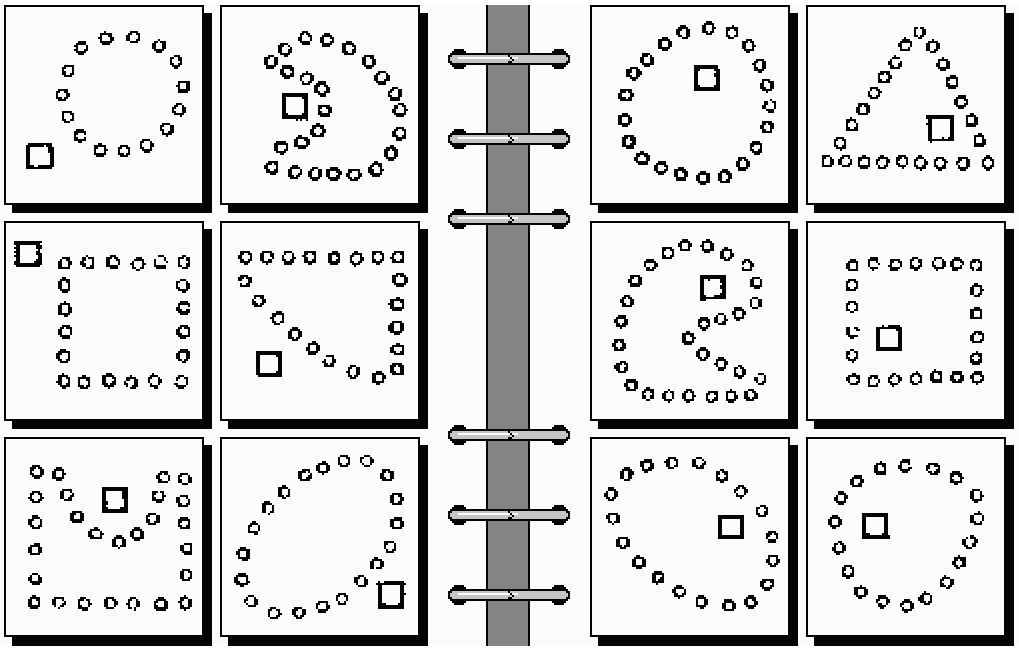


Figure 7.13: BP #84, where objects are lined up forming larger shapes

Another question is whether it is the barycenter of the object itself that is perceived, or the barycenter of its convex hull. Consider the object in Figure 7.14.

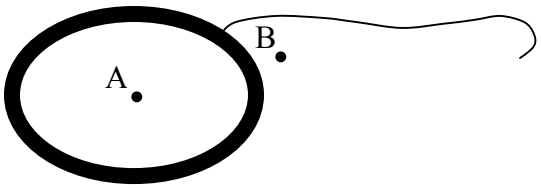


Figure 7.14: Barycenter of object (A), and barycenter of convex hull (B)

Figure 7.14 shows a rather extreme case where the barycenters of the object (A) and its convex hull (B) are widely separated. The object consists of a heavy

"body" and a lightweight "flagellum". In lining up this object with other objects, it seems more natural to use A than B. Phaeaco would use A, although it is in a position to perceive the barycenter of a convex hull in a suitable context. Notice that the barycenter node in Figure 7.11 is connected directly to the object node, which implies that this is the barycenter of the object itself.

The next figure shows the final representation of the Λ-shaped object.

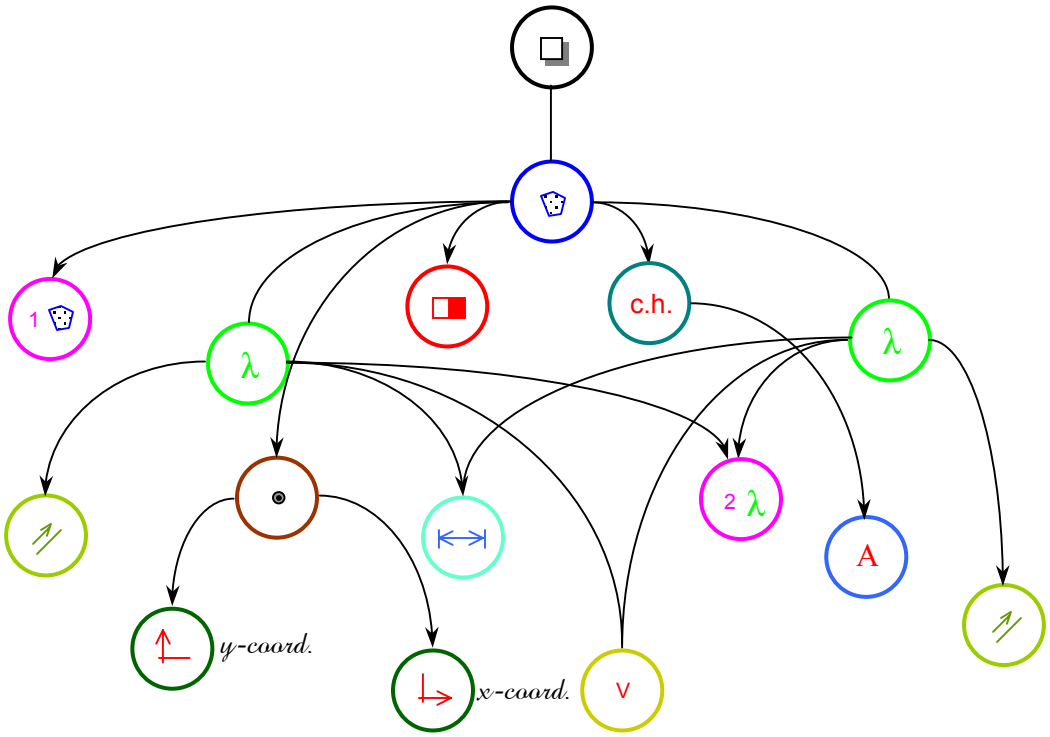


Figure 7.15: Final representation, with coordinates of barycenter

Once the barycenter is perceived, it places two codelets on the coderack, one each for its *x*- and *y*-coordinate. These codelets are almost always selected and run once on the coderack, and thus Phaeaco gets an idea of "where" this object is within the visual box. Finally, the area of the convex hull has been added in

Figure 7.15. Although the object consists of merely two line segments, the area of the convex hull gives a rough idea about how large an object these lines make.

This concludes the quick tour of representation-building, based on the example of the Λ-shaped object of Figure 7.1. As was mentioned earlier, a representation might not include all of the nodes shown in Figure 7.15, or it might include a few more (such as the angle between the two lines as a feature of the vertex, the coordinates of the vertex and/or the free end-points of the two lines, and so on). What is represented at any given look at the input is probabilistic. However, the probabilities are not completely random, but biased. For example, it is impossible for Phaeaco to fail to represent the two line segments, or the fact that they meet each other and form an object. But as we proceed deeper into the structure of Figure 7.15 (starting from the box-node at the top and proceeding to further linked nodes), the possibility exists that some of the less important leaves in this structure will not be built into it. In a different look at the input, however, such leaves might be included, and others omitted. The importance of the various representational elements is determined by the urgency of the codelets.

At this point the reader might expect a precise list of codelets generated by each type of representational node. However, this level of detail will not be provided, because the exact number, quality, and urgencies of generated codelets per node type is *not* what makes the system work. Different implementations might cause different codelets to be put on the coderack, selecting and running them at various times due to their varied urgencies. The important proposal here is that the general outline of representation-building be followed, as described in this and subsequent chapters. For the same reason, not even a precise listing of all visual primitives is absolutely crucial, though all those that Phaeaco's current implementation uses will be discussed in §7.4.

7.2 Activation of nodes and monitoring activity

A detail that was glossed over in §7.1 was the issue of termination of the representation-building activity. How were the codelets "persuaded" to cease building further elements in the structure, thereby avoiding the perception of details[36] the human eye almost never sees? One might assume that all generated codelets were eventually given their chance to run, and when the coderack was left empty, activity ended. This cannot be the case, however, because, as was mentioned in the paragraphs immediately following Figure 7.3, each λ-node automatically places on the coderack six codelets, one of which "wants" to measure the width of the line segment, another that "wants" to represent the midpoint, and so on. Why were those codelets not selected? Even if their urgencies were very low, when they were left as the only choices on the coderack they should have been given a chance to run. An explanation is in order for how the building activity might end while the coderack is still not empty.

The answer to this question involves an additional element of the structure of a node, which might seem insignificant at this point but will later play an important role in the description of the properties of the LTM. The additional element contained in each node is called *activation,* and is implemented as a real number in the interval (0, 1).[37] Activations can be increased in discrete steps by small "injections", as we shall see, and they decrease gradually and automatically also in discrete steps, as time goes by. Before specifying the exact parameters that

---

[36] Examples of such details would be the bisector of the angle, the slope of that bisector, the mid-points of the two line segments, the line that connects those mid-points, the line that connects the two endpoints — thus completing the triangle — and many more.

[37] The parentheses (as opposed to brackets) are intentional, and mean that the two end-values of the interval, 0 and 1, are excluded as possible values of an activation, as we shall soon see.

control changes in value of an activation, let us see how activations are used in the build-up of a representation.

Consider again Figure 7.5, repeated and enriched below as Figure 7.16.

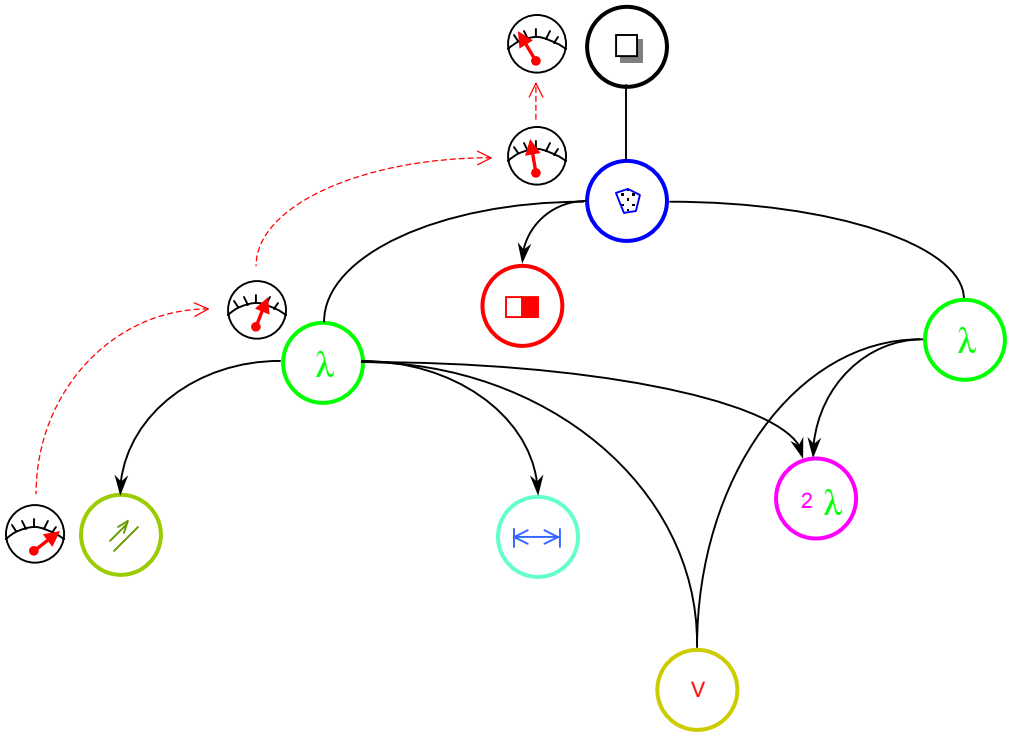


Figure 7.16: Figure 7.5, repeated, with activations on some nodes illustrated

When the leftmost node, representing a slope, was added to this structure, its activation was assigned the maximum value, a number very close to 1 (we shall soon see what that is). Immediately, this injected a bit of activation into its parent, the λ-node (and in general to all of its parents, if it had more than one). In turn, the λ-node injected a bit of an upward push to the activation of its own parent, the object-node; and so on, all the way to the ultimate ancestor, the box-node. This situation is depicted in Figure 7.16 with a "gauge" next to the nodes mentioned, the reading of which is assumed to be high if the activation is close to 1. The

reading of the λ-node appears to be slightly lower than that of the newly introduced slope-node in the figure, because some time has passed since the introduction of the λ-node into the structure (a time at which its activation started at its highest value), and, as was mentioned earlier, activations drop gradually as time goes by, if nothing else happens to them. Naturally, the newly introduced slope-node gave a small upward push to the activation of the λ-node, but this was not enough to cause the latter to reach its maximum value. Similarly, the gauges of all other ancestor nodes are shown with their readings progressively lower as we move toward the root of the structure, reflecting the earlier times at which they were introduced.[38] This remark does not imply that the activation of a node will never reach high readings again. Often it happens that there is a flurry of activity in descendant nodes that, collectively, pushes the activation of a parent node to its maximum, only to have it gradually drop again later, due to the passage of time.

All this leads to an explanation for how the activity in the workspace can end while still having codelets waiting in the coderack. What does the trick is the activation of the root-node of the representation — the box-node in this case.[39] When the activation of the root-node drops below a threshold (a minimum value close to 0, also to be explained soon), the representation is considered complete, and the input considered "seen" to Phaeaco's satisfaction at this stage, forcing all further codelet-instigated processing in the visual box to cease.

A description of the mechanism according to which activations increase and decrease is now given. The reader should keep in mind that the same mechanism is used in various other components of the architecture: in "strengths" of links,

---

[38] This orderly progression is not necessary. In §7.1 a λ-node was introduced before its parent object-node. Nonetheless, Figure 7.16 attempts to convey the general idea.

[39] A box-node is not the only candidate for a root-node of a representation. The node for a BP side, for instance, contains six boxes, and the node for an entire BP contains two side-nodes.

and in the activation and "significance" of LTM nodes, all to be explained in later chapters.

The main element in an activation structure is its *value,* a real number in the open interval (0, 1), depicted on the *y*-axis of the graph in Figure 7.17.

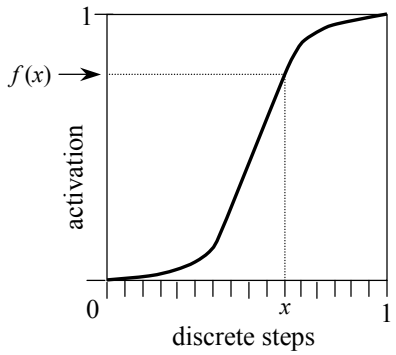


Figure 7.17: A sigmoid that shows how the activation value (on the *y*-axis) can change in time

The interpretation of the graph in Figure 7.17 is as follows. There is a quantity that can take on values only on the discrete marks depicted at regular intervals along the x-axis, the number of which is a parameter of the activation structure. Let x be the value of the marked location where the dotted vertical line is drawn. The sigmoid function *f,* also shown on the graph, defines $f(x)$, which is the value of the activation at a time during which the coordinate on the x-axis is at x.

The only way an activation can increase is by receiving a signal of the form "advance to the next marked location on the x-axis". Thus, since f is monotonically increasing, if x moves to the next spot, $f(x)$ will take on a larger value.

There is no way to ask explicitly an activation value to decrease. Activations decrease only "naturally", by the passage of time. After a given time unit passes, the quantity on the x-axis moves to the previous marked location, and $f(x)$

decreases accordingly. The time unit required for this to happen is another parameter of the activation structure.

Why must the shape of *f* be a sigmoid? Would other shapes work as well? First, observe that a strictly monotonically increasing function that starts at 0 and ends at 1 is needed, so that, given the abstract signal "intensify activation", $f(x)$ will actually increase, not decrease. Second, there only are a few possibilities of *simple* curves with these constraints — and the reasoning for a complex curve would be hard to justify. Figure 7.18 presents three such possibilities.

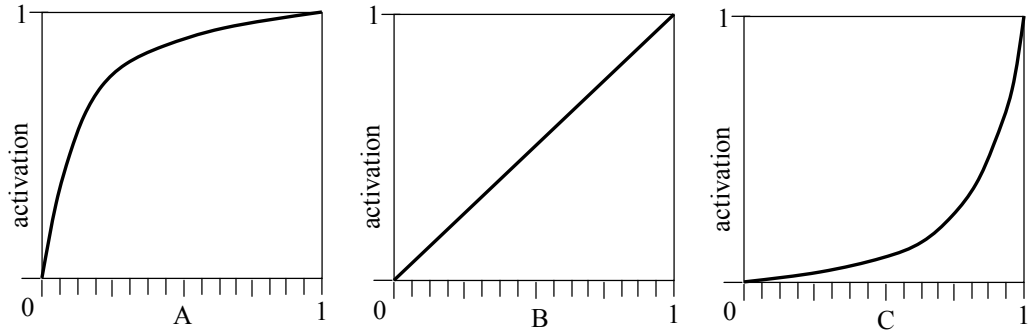


Figure 7.18: Three alternative possibilities for a monotonically increasing function *f*

The leftmost function (Figure 7.18 A) has an initial fast-rising part, and a final slow one as it approaches 1. The rapid initial rise in this function is inappropriate. An activation implements in an abstract way the idea "let's see if there is something interesting here", or "let's pay attention[40] to this", where "this" can be a node, an input box, a concept, a whole side of a BP, etc. — let us use "idea" in this discussion to refer to anything that can be activated. If the initial part of the activation function increased as rapidly as in function A, the system would pay much attention to this idea on only the tiniest suggestion that it is significant.

[40] This is a sub-cognitive "attention", not to be confused with the conscious focus of attention.

However, the opposite should in fact happen: the system should be conservative in incrementing activations at the outset, when evidence of importance is still insufficient, so as to be able later (when enough evidence arrives) to separate the more important ideas from noise. Function A would allow most of the "chaff" to pass as "wheat".

The rightmost function C, on the other hand, has a different kind of problem. If we want to allow activations to drop gradually and naturally, as time goes by, function C is problematic because it cannot hold onto any highly activated idea at all. As soon as a short time elapses, activation drops dramatically. The opposite is needed: if an idea has been identified as interesting, hence highly activated, the system should be able to hold onto it for a while; perhaps more evidence will soon arrive and confirm its importance.

Finally, function B was supplied only for the purpose of completeness. It is a "bland" function that acts neither conservatively to suppress noise, nor prudently to retain important ideas. In conclusion, the most felicitous function must be one that has the shape of a sigmoid.

The previous abstract description might not seem to make sense in the context of node activations of working memory representations. But the concept of activation is highly versatile and pervasive in Phaeaco's architecture, and since it is introduced in this subsection it is explained in all its details here, even though its full force will become evident only in the context of the discussion of LTM.

At this point the reader might infer that function f must be explicitly defined somehow in the structure of an activation. This would be a rather straightforward implementation. We could, for example, define $f(x)$ as some form of an arctangent function, $f(x) = \arctan(a \cdot x + b) + c$, which has the shape of a sigmoid. We could specify the parameters a, b, and c, so that f passes through points (0, 0)

and (1, 1), and has its point of inflection at (½, ½), as in Figure 7.17. However, sometimes the most general and straightforward solutions prove disastrous in the implementation of a complex system. Phaeaco is a highly parallel system that would benefit greatly if implemented in a computer with a true parallel design. At present, however, Phaeaco's parallel nature is constrained to run on single-processor computers. Overloading its computation of activation — repeated hundreds of thousands (if not millions) of times per BP-solving session — with the calculation of a function as demanding as an arctangent would result in a system that worked in theory, but in practice was unable to deliver the expected results. Accordingly, a sigmoid-like function is implemented in Phaeaco without making use of trigonometric or other math-intensive calculations. To this end, the sigmoid function is partitioned into three distinct pieces (see Figure 7.19).

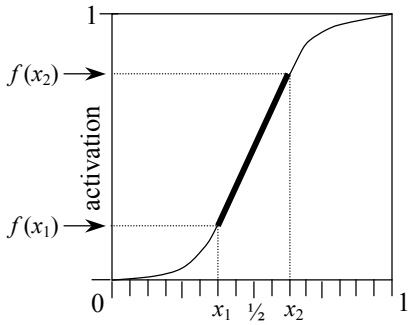


Figure 7.19: Separating the sigmoid into three constituent parts

Figure 7.19 shows a sigmoid-like function consisting of three parts: an initial curved one rising upwards, a middle linear section (highlighted), and a final curved part decelerating toward (1, 1). These three parts can be implemented in a computationally inexpensive way, by making use of only division as the most expensive operation. Specifically, the initial and final curved parts can be pieces of two hyperbolas: $f_1(x) = 1/(a_1 x + b_1) + c_1$ and $f_2(x) = 1/(a_2 x + b_2) + c_2$. It is not

necessary to insist that the sigmoid-like curve be smooth; for example, at point x_1 the slope of the tangent as $x \rightarrow x_1$ from the left can be different from the slope of the tangent as $x \rightarrow x_1$ from the right; and similarly for x_2. Other simplifying assumptions, implied by the fact that the sigmoid-like function has a center of symmetry at (½, ½), are that $x_2 = 1 - x_1$ (assuming $x_1 < ½$), and that parameters a_1, b_1, and c_1, are not independent of a_2, b_2, and c_2, since the two curved pieces are mirror images of each other, and thus pieces of the same hyperbola. But we need not become further mired in the specifics of the implementation at this point. Suffice it to say that a number of simplifying decisions, such as those above, can lead to an implementation that uses computer resources efficiently.

7.3 Numerosity

The percept of numerosity was briefly introduced in §1.2.1, and BP's that depend on the perception of numerosity were discussed in §5.1.1, along with an outline of how Phaeaco handles such BP's. The present subsection discusses in greater detail the percept of numerosity and its implementation in Phaeaco's architecture.

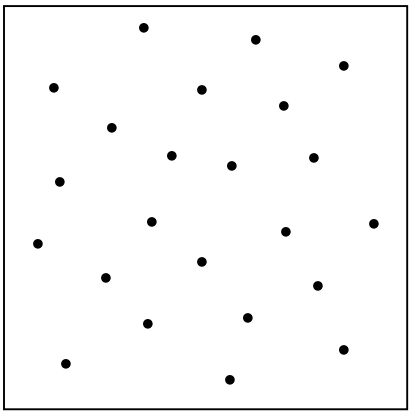


Figure 7.20: How many dots are present, without counting?

An example of the simplest kind of the percept of numerosity is given in Figure 7.20. Assuming we are allowed to look at the figure for no more than a second, how many dots are there in the box? Even though an exact answer cannot be given, people can make a rough estimate. Clearly, hardly anyone would report fewer than 10 or more than 50 dots (and these limits seem far too conservative). Reasonable guesses could be in the range between 15 and 30.

7.3.1 Background

That we do not need to resort to explicit counting to have a rough sense of the quantity of something is a well established finding in psychology. Since the 1920's, psychologists have been examining the relation between the number of occurrences of an input feature ("stimulus") and the strength of its association with a corresponding action ("response") by a cognitive agent (Thurston, 1927). Because the estimation of the absolute number of a percept (such as the number of dots in Figure 7.20) depends on the subject's prior experience with small and large numbers, and even on their cultural background, later studies focused on the perception of ratios, or differences of numbers.[41] For example, in one study subjects were presented with 20×20 arrays of short vertical and horizontal lines, and were asked to estimate the proportion of one of the two orientations in the array (Shuford, 1961). It was found that the estimates were more accurate if the actual proportion of the specified target-orientation was either small (20%–30%) or large (70%–80%), whereas the worst estimates were made when the proportion was around 50%. This is consistent with the idea that smaller numbers are

[41] However, studies letting subjects report directly a number to describe the perception of numerosity have not been absent (e.g., van Oeffelen and Vos, 1982).

perceived more accurately than larger ones.[42] The study included also red and blue squares instead of vertical and horizontal lines, and the results were similar.

Other studies focused on response times. It was found that subjects are faster in determining the correct order of two digits if the difference between the two digits is large, rather than small (Moyer and Landauer, 1967). Experiments were repeated with letters of the alphabet, judging their alphabetic distance (Parkman, 1971), dot patterns (Buckley and Gillman, 1974), rows of dots, and the auditory form of spoken English words for numbers (Shepard, Kilpatric *et al.*, 1975). In all cases the decision time was found to follow approximately a logarithmic function of the numerical difference between the two compared quantities, known as Welford's formula (Moyer and Landauer, 1973; Welford, 1960).

$$RT = a + k \cdot \log\left(\frac{L}{L - S}\right)$$

Equation 7.1: Welford's formula for reaction time in numerosity comparison

In Welford's formula (Equation 7.1), which is an elaboration of the Weber–Fechner law,[43] RT stands for "reaction time", L and S are the larger and smaller of the two compared quantities, respectively, and a and k are constants.

In addition to people being able to compare quantities, it is well known that animals also have a sense of numerosity that varies from rudimentary to astonishing, depending on the species. In a series of studies, experimenters taught hungry rats to press levers a number of times to receive food. The rats learned by trial and error (by hitting the levers randomly) that, for example, after four hits on

---

[42] A large number of horizontal lines in an array implies a small number of vertical ones. Thus, in seeking to estimate the proportion of one kind in a sample that consists of two kinds, "small" means close to one of the two ends (0% or 100%), whereas "large" means close to 50%.

[43] According to the Weber–Fechner law, linear increments in sensation S are proportional to the logarithm of stimulus magnitude m: $S = k \cdot log(m)$ (Fechner, 1860; Weber, 1850).

lever A and a final hit on lever B, the door of a compartment with food would open; other rats learned that the "right" number on lever A was eight; and so on, up to such "exotic" numbers as 12 and 16 (Mechner, 1958; Platt and Johnson, 1971). It turns out that the performance of the rats is very telling regarding their perception of numerosity. The animals never learned to hit lever A the right number of times accurately, but only approximately. When their attempts were cumulatively plotted on a graph, it was found that the population of hits approximated a Gaussian with mean value close to the right number.

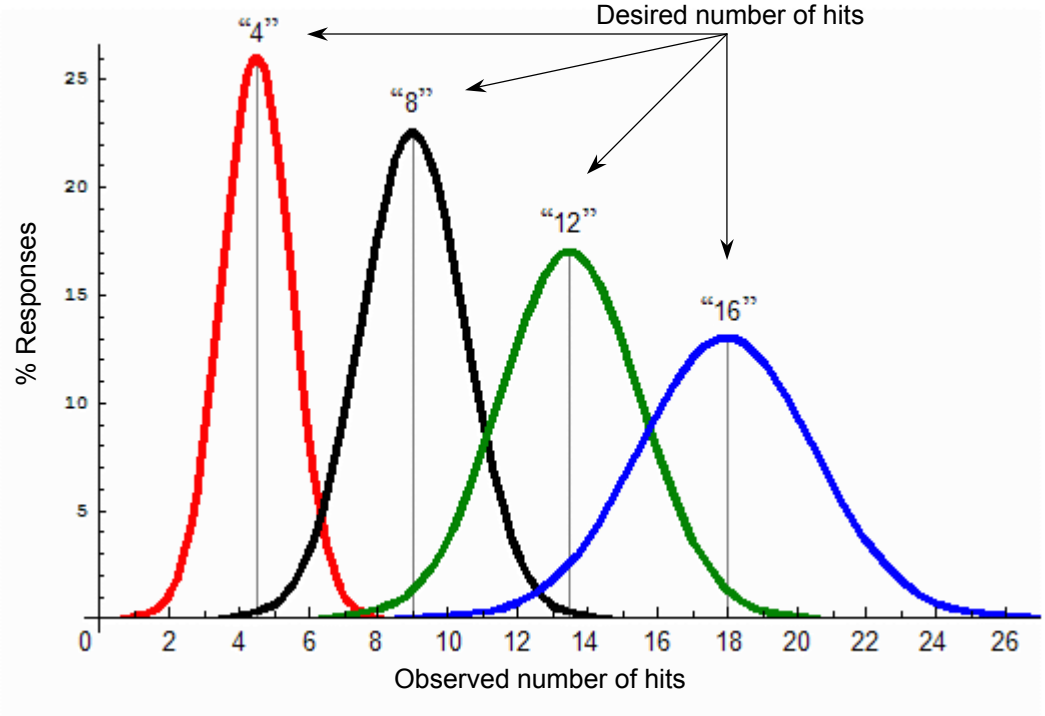


Figure 7.21: Rat numerosity performance (adapted from Dehaene, 1997)

Figure 7.21 depicts an idealization of the rats' performance. (For the actual data, see Mechner, 1958.) The depicted curves would be more accurate if they were slightly skewed towards the left, especially those at the low-numerosity end.

Even so, it is clear from the figure that the rats overestimated the desired number of hits, and the higher the desired number, the larger their overestimation. This was probably an artifact of the experimental procedure: the rats received a penalty for switching to lever B prematurely, after an insufficient number of hits on A. (Without the penalty they would immediately try to hit on B.) Also note that their accuracy dropped as the desired number of hits increased: the variability of their responses increased in proportion to the number that the rats were aiming for.

Variations of such experiments showed that not only rats but many other species are capable of perceiving numerosity in a variety of input forms: food items, sounds, time duration, light flashes, and more. Raccoons, for instance, can learn to select the transparent box that contains exactly three grapes, and to ignore similar boxes that contain two or four grapes. Birds can be taught to pick the fifth seed they find when visiting several interconnected cages. Pigeons, in particular, can discriminate between forty-five and fifty pecks at a target, under some circumstances (Dehaene, Dehaene-Lambertz *et al.*, 1998).

Further support for animal arithmetic skills comes from findings such as that chimpanzees are capable of integer and even fraction addition. In one experiment, a chimp was allowed to select one among two trays with piles of chocolate chips for eating. Tray A contained two piles, one with four and another with three chips. Tray B contained also two piles, one with five, and another with a single chip. The two piles were widely separated in each tray. After watching the situation carefully, without any prior training, the chimp selected the tray with the 4+3=7 chips, instead of the one with 5+1=6 chips (Rumbaugh, Savage-Rumbaugh *et al.*, 1987). In another experiment, a chimpanzee was first trained in fractions. When presented with a glass half-filled with a blue liquid, the animal had to point to an identically filled glass standing next to one that was three-quarters full. Then the

task was abstracted to one in which, after being shown the half-filled glass again, the animal had to make a choice among a half apple or three-quarters of an apple, which the chimp passed successfully. Finally, the stimulus consisted of a half-filled glass and one-quarter of an apple, whereas the choice was between one full disc and a three-quarters disc. The animal chose the latter more often than chance alone would predict. Whatever underlying representation was used, the chimp must have performed the equivalent of ½ + ¼ = ¾ in human math notation (Woodruff and Premack, 1981).

If animals are capable of such arithmetic feats, it is hardly surprising that human babies have similar abilities. Indeed, Piaget's constructivist theory notwithstanding,[44] perception of numerosity has been confirmed in infants. Introducing the now widely used method of infant habituation[45] in the 1980's, Prentice Starkey first established that children between 16 and 30 weeks of age were able to discriminate between small numbers, such as two and three (Starkey and Cooper, 1980). Later it was argued that even newborns could discriminate between numbers two and three a few days after birth (Antell and Keating, 1983). Several more experiments have established that infant abilities in numerosity perception are as sophisticated as those of other species. Indeed, no adult

---

[44] Jean Piaget, in mid-twentieth century, claimed that children are born with a mind that is essentially "blank" in mathematical abilities, and gradually reach the abstract concept of number at the age of six or seven, after first having been trained in more fundamental notions, such as sensory-motor skills, the elements of logic, and the ordering of sets; before that age, the child is simply not "ready" for arithmetic (Piaget, 1952; 1954). Today Piaget's theory is known to have misinterpreted a number of results from early experiments, and to contradict more recent experimental findings (Dehaene, 1997; Mehler and Bever, 1967).

[45] According to this method, a child is shown repeatedly scenes that include identical, or very similar percepts, until the child is habituated and looks away very soon after each presentation. When a perceptually different scene is introduced without warning, the child's fixation time on the "interesting" scene is recorded. A longer fixation time indicates the child noticed the difference.

chimpanzee or other animal seems capable of competing with human children older than three years of age (for a review, see, e.g., Lakoff and Núñez, 2000).

The perception of numerosity by humans (of any age) is different from the concept of "number", acquired after years of formal training. Though we instantly have a sense of quantity by looking briefly at the dots in Figure 7.20, we feel at a loss if asked to report their exact number. Nonetheless, given sufficient time, we can employ some method for *counting* the dots (perhaps using our fingers as aids to conceptually group and avoid re-counting some already-counted dots), by which we can report the exact number (23 in Figure 7.20). In many aspects, our numerosity perception does not differ at all from the corresponding ability of some animals. Specifically:

- our ability to discriminate between quantities is sensitive to the difference of those quantities: it is easier to discriminate between 5 and 10 than between 5 and 6; and
- our discrimination ability is also sensitive to the absolute magnitude of the compared quantities: it is easier to discriminate between 5 and 6 than between 25 and 26.

Both observations are predicted by Welford's formula (Equation 7.1).

7.3.2 The accumulator metaphor

What cognitive mechanism could possibly account for the arithmetic abilities of animals, or for something like Welford's formula? Iterative (algorithmic) mechanisms have been proposed by some authors (Buckley and Gillman, 1974; Moyer and Bayer, 1976), but they apply only to comparisons of quantities, and not to perception of quantity *per se*. Instead, Phaeaco implements a mechanism

that models directly the perception of quantity, called "the accumulator metaphor" (Dehaene, 1997). Due to its simplicity, this model is particularly elegant.

According to the accumulator model, each of the dots in Figure 7.20 adds to an internal cognitive "accumulator" a quantity that is not exactly "1", but "around 1" (because in biology this is implemented with an inexact chemical quantity, rather than digitally). The added quantity can be thought of as a random variable from a Gaussian distribution with mean $\mu = 1$ and standard deviation σ_0 — a constant for each individual, but varying slightly across individuals. Thus, for example, the 23 dots of Figure 7.20 add to the accumulator 23 random numbers generated from a normal distribution N (1, σ_0), for some σ_0. The larger the number to be perceived, the larger the margin for error in the accumulator. Does this model explain the observations of §7.3.1 regarding the perception of numerosity?

The sampling distribution of the sum of n Gaussians N (μ_i, σ_i), $i = 1, \ldots n$, is again a Gaussian N (μ_Σ, σ_Σ) (Equation 7.2; this can be proved by induction on n).

$$\mu_\Sigma = \sum_{i=1}^{n} \mu_i \qquad \sigma_\Sigma = \sqrt{\sum_{i=1}^{n} \sigma_i^2}$$

(a) (b)

Equation 7.2: (a) mean and (b) standard deviation of a sum of Gaussians

In the case where all μ_i are equal to 1 and all σ_i are equal to a constant σ_0 it follows that $\mu_\Sigma = n$, and $\sigma_\Sigma = \sqrt{n}\,\sigma_0$. Thus, the probability density function G_n for the n-th cumulative Gaussian is given by Equation 7.3.

$$G_n(x) = \frac{1}{\sqrt{2\pi n}\,\sigma_0} e^{-\frac{(x-n)^2}{2n\,\sigma_0^2}}$$

Equation 7.3: Probability density function for a sum of Gaussians N (1, σ_0)

Phaeaco uses Equation 7.3 both for the task of estimating the quantity of anything perceivable and for the task of comparing quantities as follows:

Suppose the input contains n similar percepts (they can be objects, slopes, angles, relations, or anything discrete, hence countable). To form a representation of the numerosity of such percepts, Phaeaco executes the procedure suggested by the accumulator metaphor, i.e., adds a random number generated from N (1, σ_0) to a numerosity node n times. (Such additions happen not all at once, but whenever one of the instances of the counted entity is perceived — cf. also the construction of the node that counts the numerosity of the two lines in the example of §7.1.) When Phaeaco is asked to report this numerosity as an integer number, it outputs the integer that is closest to the accumulated real-valued quantity. For small numbers (up to 5), the reported integer is almost always accurate; but for larger numbers errors accumulate, and the probability of reporting the wrong integer increases, just as would be expected from a biological cognitive agent.

It follows from the previous paragraph that to calculate numerosity, Phaeaco uses Equation 7.3 only implicitly. In contrast, when comparing two discrete quantities, the equation is used explicitly. Specifically, to compare two numerosity nodes with values L and S, Phaeaco executes the following iterative algorithm.

1. The probability density functions $G_L(x)$ and $G_S(x)$ are considered. Two initial samples S_L and S_S of size m each are generated from $G_L(x)$ and $G_S(x)$, respectively, where m is a small constant (e.g., reasonable values are between 10 and 15). At this point the reaction time RT is set to m.
2. A standard statistical decision test is applied to determine whether the samples S_L and S_S originate from different populations. The test uses the

known standard deviations of the populations, $\sigma_L = \sqrt{L}\,\sigma_0$ and $\sigma_S = \sqrt{S}\,\sigma_0$, respectively, and hence the observation that the random variable

$$z = \frac{(\bar{x}_1 - \bar{x}_2) - (L - S)}{\sqrt{\sigma_L^2/m' + \sigma_S^2/m'}} = \frac{(\bar{x}_1 - \bar{x}_2) - (L - S)}{\sigma_0\sqrt{(L + S)/m'}}$$

is a standard normally distributed variable, where m' is the current size of the samples S_L and S_S (initially equal to m). (Cf. also §8.2.2.)

3. If the test determines that the populations differ, the algorithm ends.
4. If the test cannot determine a difference with sufficient confidence, one more sample value is added to each of S_L and S_S, the reaction time RT is incremented by one, and the algorithm returns to step 2.

Simulation can show that the above procedure yields reaction times that follow the general trend of RT in Welford's formula. The constant a in Equation 7.1 is akin to the constant m in the algorithm above. Nonetheless, some words of caution are necessary.

Welford's formula should not be regarded as a "law" to be applied blindly in numerosity perception. Specifically, suppose $S = 1000$ and $L = 1001$ in a display of two boxes with the corresponding numbers of dots. It is then essentially impossible for the human visual system to discern any difference in numerosity. Welford's formula predicts that an answer will be given on average in $a + k \cdot 7.9$ seconds, but it does not account for the fact that this answer will be random. To address this problem, Phaeaco uses an upper bound for the number of cycles in the algorithm above. If this upper bound is exceeded, the algorithm concludes that the numerosity appears about equal in both cases.

The consistency of the accumulator metaphor (as implemented in Phaeaco) with experimental results (such as Welford's formula) should not be construed as

evidence that animal cognition also performs statistical tests of difference of two populations. Perhaps animal cognition implements the equivalent of such tests in an analogous, neuronal way. The consistency simply lends to Phaeaco a more human-like, less computer-like behavior when comparing quantities.

Finally, the perception of numerosity is a complex task — definitely more complex than examples with dots in boxes suggest. For example, it is known that when features are spatially clustered, people systematically overestimate their numerosity (Goldstone, 1993). Also, in idealized laboratory inputs that contain dots, lines, etc., it is not easy to tell whether people attend to the number or the spatial separation (density) of the counted entities. Phaeaco's treatment of numerosity should be seen merely as a step in the right cognitive direction.

7.4 Other visual primitives

Some of the visual primitives that are available to Phaeaco in its current implementation were discussed in §7.1. Several other primitives that occur in inputs more complex than a Λ-shaped object are discussed in this section.

7.4.1 Dots, points, abstract percepts, and conceptual hierarchies

If a collection of connected pixels is so physically small that even the magnification procedure[46] will fail to assign a shape to it, Phaeaco perceives the collection as a "dot". Dots can be as small as a single pixel, or as large as a region approximately 5 x 5 pixels in size, but isolated pixels occasionally can be missed altogether ("not seen") by the retinal-level image-processing algorithms. As the number of connected pixels that form the dot increases, the probability of missing the collection decreases sharply, becoming practically zero for collections of four

[46] Outlined in §5.1.5, and to be further discussed in §10.3.17.

or more pixels. Pixels that form "dots" might even have tiny holes in them (e.g., a few pixels missing), which also fall below Phaeaco's discrimination ability.

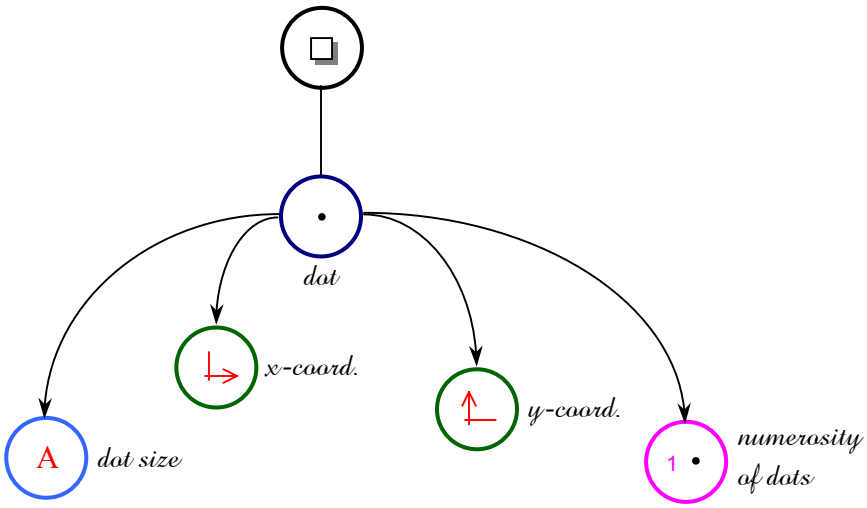


Figure 7.22: Representation of a single dot in a box

Figure 7.22 shows the representation formed for a single dot in a visual box. Besides nodes for the box itself and the dot, there are nodes representing the size of the dot, its *x*- and *y*-coordinates, and its numerosity ("one dot in the box").

The phrasing "approximately 5 × 5 pixels in size" earlier in this subsection is worthy of a generalization: no parameter in Phaeaco's architecture is a true "constant", hardwired into the code. All parameters that represent decision thresholds (such as, "Decide whether to apply the magnification procedure" in this case) are normal distributions $N(\mu, \sigma)$, where μ corresponds to the traditional value of a constant, and σ is very small. Decisions that depend on such parameters are probabilistic, taken after generating a random number r from the distribution $N(\mu, \sigma)$, outputting "yes" if $r > \mu$, and "no" otherwise.

Another generalization concerns Phaeaco's ontology of concepts. Though a distinction was already drawn between "real" and "imaginary" percepts in §7.1, there is a further class of abstract concepts that do not correspond to any percept in the input. In the present context it is appropriate to mention the notion of a "point". A "point" is to a "dot" (and to a "barycenter", see §7.1) roughly what the class "mammal" is to the class "dog" (and an imagined mammal, such as "unicorn", would correspond to "barycenter" in this analogy). Phaeaco never creates nodes in the Workspace to represent points, but there is a permanent node in LTM that corresponds to the Platonic notion of a "point", which is also connected with the Platonic notions of "dot" and "barycenter" through two-way links of type "is a kind of" (from "dot" and "barycenter" to "point") and "has subclass" (from "point" to "dot" and "barycenter").[47] Each property of a point is also inherited by its subclasses. For this reason, when Phaeaco determines that something in the input is a dot and links it to the LTM concept of a dot, it "knows" that it can create codelets for its *x*- and *y*-coordinates not because the Platonic dot is linked explicitly to such notions, but because every dot "is a kind of" point, and points are known to have coordinates. Nonetheless, dots have the additional property of size, which points (and barycenters) lack.

This, of course, is none other than the classical (Aristotelian) notion of a hierarchy of categories, which object-oriented programming languages reinvented and implemented as "class hierarchies". Phaeaco implements this notion in its conceptual network in LTM (more in chapter 9).

---

[47] A figure depicting the represented concepts is deferred until chapter 9 where the LTM is introduced, so as to avoid including representations of a completely different nature in the present chapter.

7.4.2 Vertices, Touches, Crosses, and K-points

When two or more lines (straight or curved) share a point, they can do so in more complex ways than the simple "vertex" introduced in §7.1. Consider straight lines only, for the sake of simplicity.

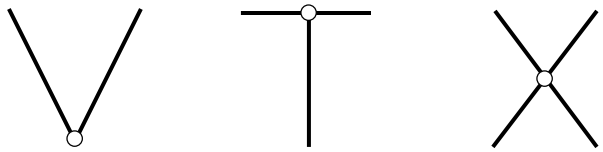


Figure 7.23: Three ways in which two lines can meet: a vertex, a touch-point, and a cross

The three possible ways in which two line segments can meet are shown in Figure 7.23: the lines can meet at a vertex; or one line can touch the other at a "touch-point"; or they can cross each other at a "cross-point". Conveniently, the shape of the Roman letters V, T, and X serves as a mnemonic of the words "vertex", "touch", and "cross". Phaeaco uses the symbols shown in Figure 7.24 to depict such points in representations.

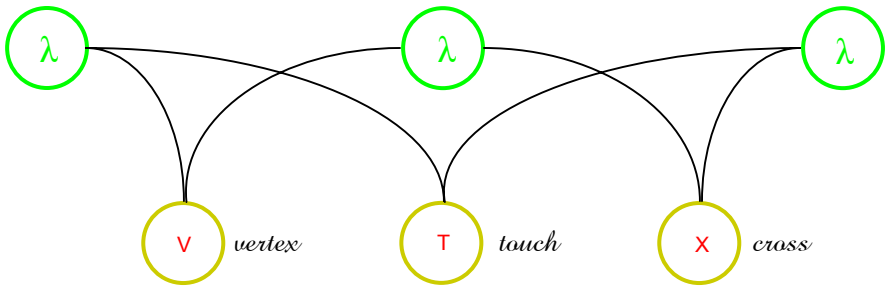


Figure 7.24: Representation of V, T, and X in Phaeaco

If more than two line segments are involved, then more complex structures can form, as shown in Figure 7.25. All such cases will be collectively known as

K-points (using the shape of the letter "K", or perhaps "komplex", as a mnemonic).

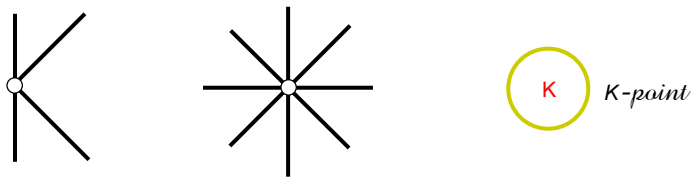


Figure 7.25: More complex intersections: K-like (a), star-like (b), and their representation (c)

A question that arises is how to represent K-points. Clearly, each such point admits a multitude of different "views" (see also §7.4.11) of how the lines are related. For instance, the K-point in structure (a) in Figure 7.25 can be seen as two touch points that coincide (the two slanted lines touching the vertical one); but also as a vertex (formed by the two slanted lines) that lies on a vertical line; or as two vertices that coincide (formed by a V-like and a Λ-like shape); or in a variety of other ways. Similarly, the star-like structure (b) in the same figure suggests that the number of possible descriptions grows exponentially with the number of lines that participate in the formation of the point.

Phaeaco answers this question by neglecting the detailed and different ways in which lines at K-points are related, opting to keep a summary only of the structure. For example, structure (a) in Figure 7.25 will cause the creation of a K-point node (c) and three λ-nodes linked to it: one for the vertical line segment, and two for the slanted ones. The various ways in which these lines are connected at the K-point is not important as a first representational approximation; if needed, the structure can be examined more carefully later, and complex connections can be "deduced" in a logical and systematic manner.

Structures made of lines such as those in Figure 7.25 suggest that even a simple touch point or a cross can be re-parsed and seen as consisting of more than two lines. Indeed, there is a BP that exploits precisely this idea (Figure 7.26).

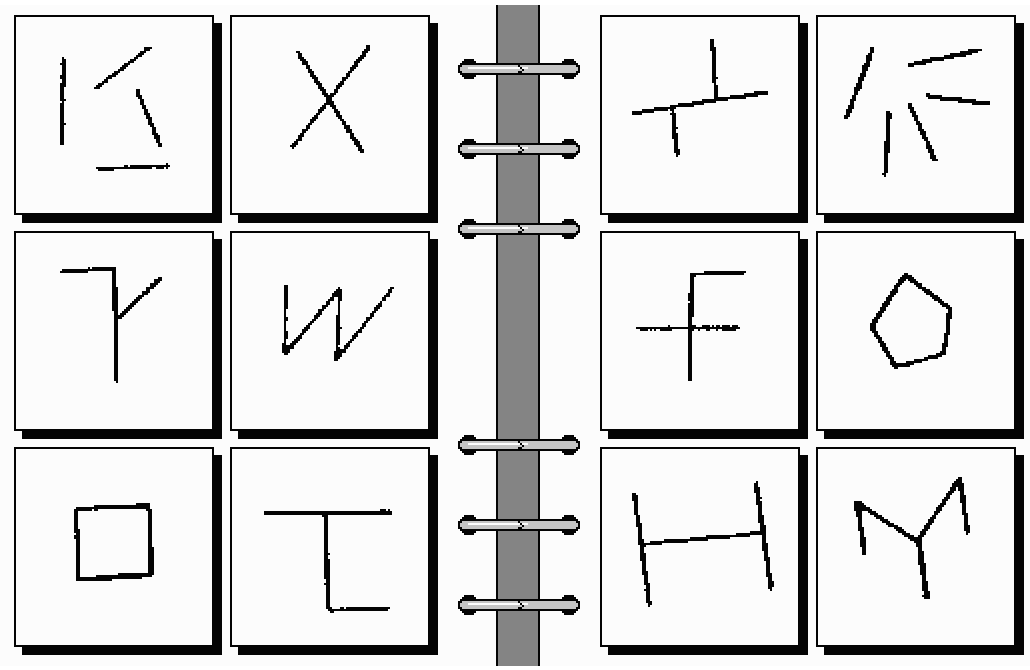


Figure 7.26: BP #87, necessitating a re-parsing of the intersected lines

BP #87 (Figure 7.26), is a problem that looks simple, but is tricky. At first, the object in box I-B is perceived as an X. Similarly, there appear to be W-like, T-like, F-like, O-like, H-like, and M-like objects in other boxes. All these are distractors, because by registering them as "letters" we are led away from the solution. In competition with this idea is the idea of "four lines", which is most typically brought into conscious focus after one pays attention to the four isolated lines in box I-A, and is reinforced by the four lines making up the square and the W-like shape. Seeking confirmation of this idea, one builds up subcognitive pressure to "break up" the X-like structure (or any of the others that might require

re-parsing) into smaller constituent lines, so that the idea "four lines" is forced onto it. Once one succeeds with one of those structures, it gets easier to apply the same idea of "re-parsing" to the rest of the objects, and to do so on the right side as well, where it yields "five lines".

BP #87 is the example that Alexandre Linhares uses to illustrate what he calls "multi-perception": the ability to break down already perceived structures in order to build new ones under some cognitive pressure (Linhares, 2000). The same idea is part of the notion of mental fluidity, a fundamental concept in other FARG projects such as Letter Spirit (Rehling, 2001), Copycat (Mitchell, 1990), and Metacat (Marshall, 1999) (see also §6.2, and Hofstadter, 1995a pp. 206-208).

The constituents of V, T, X, and K-type intersection points do not have to be straight lines. Phaeaco can represent curves, too (§7.4.5), so that instead of λ-nodes there can be nodes representing curves that are connected with intersection points in various ways.

Finally, intersection points are all "points" in the abstract sense, therefore they create codelets that "want" to find their *x*- and *y*-coordinates. But in addition, this type of point results in the measurement of *angles,* which is the topic of the next subsection.

7.4.3 Angles

Angles are perceived only when an angle-measuring codelet generated by a V, T, X, or K-point is given its turn and runs. Angles are continuous features, and as such they have the statistical structure shown in Table 7.1. Thus, an "angle" in the context of Phaeaco's architecture should not be confused with the common notion of an angle, which usually includes two lines and their point of intersection; the latter structure corresponds better to Phaeaco's vertices, touches, crosses, and K-points.

Phaeaco is able to perceive only angles less than 180°, which implies that a vertex has only a single angle to be measured. But touch points and crosses have two possible values of angles that can be measured: either an acute and an obtuse angle, or two right angles (Figure 7.27).

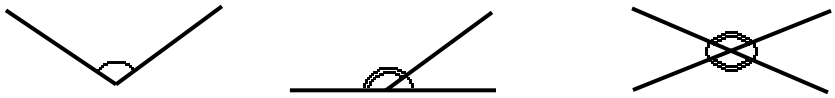


Figure 7.27: Types of angles produced by intersections of lines

Phaeaco perceives both angles of touch points and crosses, and in the latter case it registers that each angle value appears twice (Figure 7.28).

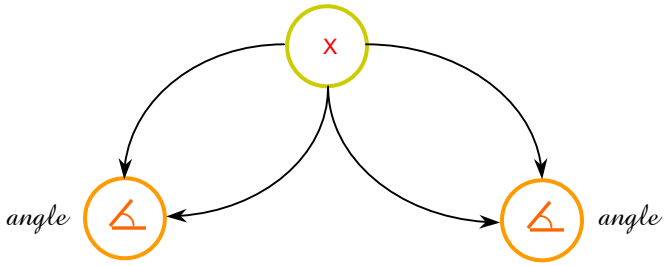


Figure 7.28: Representation of the two angles of a cross, registered (i.e., linked) twice

K-points can have many perceivable angles. In such cases, it is much more likely that Phaeaco will notice the angle formed by adjacent line segments than an angle formed by line segments that include a third one between them.

7.4.4 Line strings

When several line segments are joined together vertex-to-vertex, or through more complex intersections, we do not perceive merely a set of lines plus their intersections, but a more complex shape, which appears to be more than the sum of its parts.

For example, the shape shown in Figure 7.29 seems to have more properties than its constituents perceived in isolation: it appears to consist of a "salient frame" intersected by "minor" line segments (the latter are shown in gray in the figure to depict the salient frame properly, but they would appear as normal black lines in a BP). The salient frame ends at two end-points (marked on the figure) that seem to be more amenable to perception than other end-points of the shape, by virtue of being the two end-points of the salient frame.

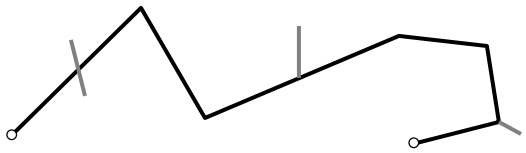


Figure 7.29: A "line string" (or possibly several of them)

Overall, the perceived frame gives the impression of a "curve", but it cannot earn the status of a proper curve. (Proper curves are discussed in the next subsection.) Such structures are termed "line strings" in Phaeaco, and, in contrast to curves, they can be considered as the geometric analogue of sequences of integers, as opposed to smooth analytic functions.

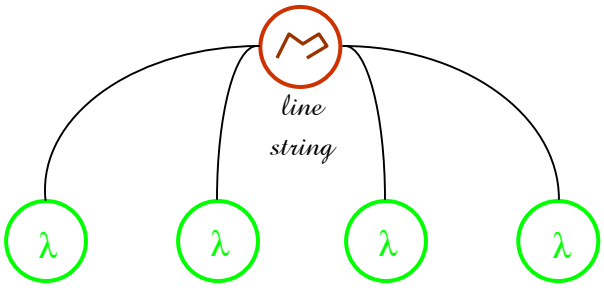


Figure 7.30: Part of the representation of a line string

Figure 7.30 shows — by necessity — only part of the representation of a line string. Besides the depicted λ-nodes and the string-node at the top, there will be nodes representing the vertices at which the line segments meet, the numerosity of the lines, and possibly a few other percepts. However, the presence of a line string suppresses the codelets generated by the λ-nodes, by lowering their urgency. This, after all, is one of the main effects of a line string in a figure: its perception as a whole "quells" the perception of the details of its constituents. Similarly, the identification of an object as a tree in reality leads us (evidently subconsciously) to reduce the amount of attention paid to its branches, twigs, and leaves, at the moment the idea "tree" is accessed in our memory.

There are several qualities that line strings share with curves. For example, they can be mentally "traced" from one end to the other, and various conclusions can be made about the manner in which the imaginary "point" traces the frame of the string. Also, line strings can form "bays" (as in Figure 7.29), or interior regions (§7.4.7) by intersecting themselves. They can also form polygons, and this is exactly how Phaeaco "understands" polygons: as closed line strings with a single interior. Although Phaeaco does not have the notion of "polygon" as one of its primitives, it is in a position to learn it by being repeatedly exposed to examples of polygonal shapes.

A few other important issues must be mentioned before leaving the subject of line strings. First, when is a line string perceived? Is a triangle a line string? Phaeaco's approach (which might by now be emerging as a pattern in the reader's mind) is that the answer is probabilistic. A low numerosity of line segments has low probability of resulting in the perception of a line string. A triangle, in particular, has almost zero probability of being seen as a line string, but as the number of lines increases above four, the probability becomes significant.

Second, what happens in situations in which there is not a single salient line string to be perceived, but more than one that could be seen as a candidate? The drawing in Figure 7.31 depicts an extreme example of such a situation.

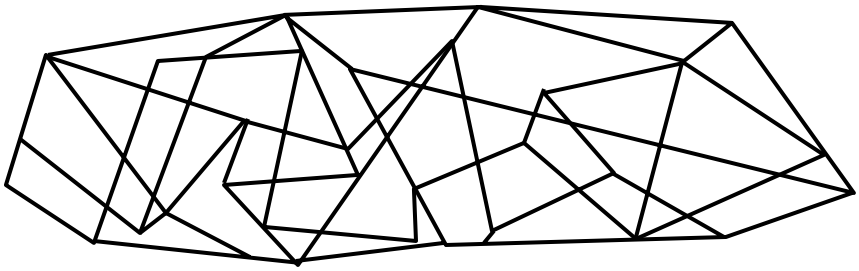


Figure 7.31: A futile exercise for computers: how many line strings are there in this "crystal"?

The last thing that can be expected from Phaeaco is that it will get tangled in a search (of anything) that leads to an exponential explosion. Simply put, there are no exhaustive search algorithms that have been implemented in the architecture. Given an object such as the one depicted in Figure 7.31, Phaeaco will indeed perceive a few line strings initially (comprising the longest line segments), but as soon as their number exceeds four or five, it will stop "trying" to perceive and represent any more of them explicitly. In general, as soon as the number of *anything* exceeds the low end of numerosity values (§7.3), a high-level codelet becomes activated in Phaeaco's Workspace, monitoring the activity of other, lower-level codelets, and suppressing them when it detects that they will attempt to work on "one more of those 'things', of which we have seen many". If, for instance, the "many things" are dots, the high-numerosity codelet will discourage the perception of the size and coordinates of each dot once a small sample of them has already been perceived; in the case of line segments, their slopes, lengths, intersections, etc., will be spared from explicit representation; and so on. In the end, given an object such as the one in Figure 7.31, a sample of its constituents

will be represented explicitly, but most will not. Instead, the focus of attention will be shifted to other percepts that do not depend on the details of its structure, such as its convex hull, area, elongatedness (§7.4.8), and so on.

Finally, it should be noted that line strings are kinds of "objects" in Phaeaco's ontology.[48] If a line string stands alone, disconnected from other shapes, its node (depicted in Figure 7.30) can replace the object-node that ordinarily would have been constructed first.[49] This is once again a general observation: identified shapes that stand by themselves replace with their specific nodes the more general nodes (e.g., "object") that were inserted in the representation before the identification occurred. If, however, the string is part of a larger structure, its node will be connected as part of an overall object node.

7.4.5 Curves

Several of the characteristics of curves were discussed in the previous subsection in the context of line strings. Indeed, before a curve is identified as such, it is first identified as a line string at the retinal level; and before a conceptual representation of a line string is made at the cognitive level, other retinal-level routines act and identify the string as a curve. The details of these algorithms will be given in §10.3.8. Here it suffices to say that in Phaeaco's ontology, both curves and line strings inherit the properties of a more abstract kind of object called a "linear structure" (for lack of a more descriptive term). A linear structure is an "object", but it has a few extra properties: it can possess two end-points, can form "bays" and interiors — thus is capable of intersecting itself — and is traceable by a point from one end to the other.

---

[48] But see also the first paragraph of "Curves" (§7.4.5) for a qualification of this statement.

[49] Recall the construction of such a node in the discussion of the Λ-shaped object in §7.1.

Curves introduce a few more properties of their own. A "bay" is formed by a piece of a curve where the points from which the curve was constructed (at the retinal level) have a local maximum density (Figure 7.32). Bays have a certain measure called "curvature" in differential geometry, which Phaeaco perceives at the retinal level by fitting a circle to the bay and noticing its center.

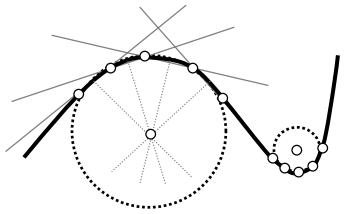


Figure 7.32: Construction of "center of curvature" in two "bays" of a curve (retinal level)

Figure 7.32 shows the way "centers of curvature" are constructed: by drawing perpendiculars to near-tangent lines of the curve, and registering the approximate point where these perpendiculars meet. Although the center is known only at the retinal level, the radius of the circle provides a measure of curvature at the cognitive level; so the latter "knows" how sharp the bay of a curve is without having access to the actual number, or to the computational method by which this measure was found. The cognitive level can only report that a curve is "sharp", "normal", "blunt", and so on, i.e., in qualitative terms.

Once the circle and the center of curvature are known, the tangent line at any point of the curve near the approximating circle can also be computed. Thus, the cognitive level is capable of "imagining" tangent lines to a curve, if necessary.

Nonetheless, not all curves are well-behaved, as the discussion thus far might imply. A *spiral,* for example, has no bays. Phaeaco is at a loss when presented with a spiral, being unable to perceive anything further than "there is a curve", possibly including its endpoints and barycenter (every curve has one, being an

"object"). Consequently, BP #16 (see Appendix A), which depends on a property of spirals (their clockwise or counter-clockwise tracing), is unapproachable by Phaeaco at present. Another concept unknown to Phaeaco in its current implementation is the "direction" of a bay (if the bay were a "satellite dish" — a parabolic mirror — it would be turned to the direction of a "satellite").

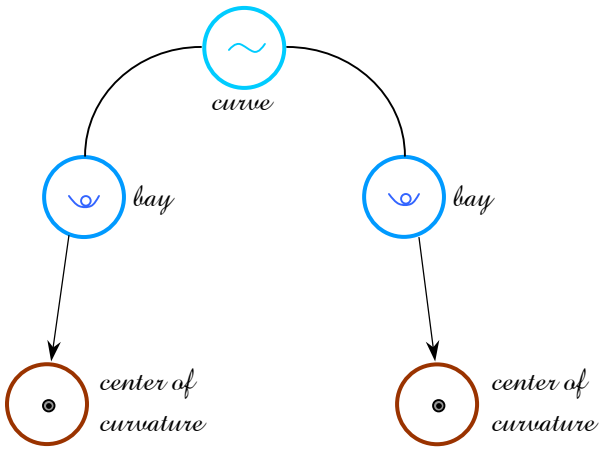


Figure 7.33: Representation of a curve with two bays and their centers of curvature

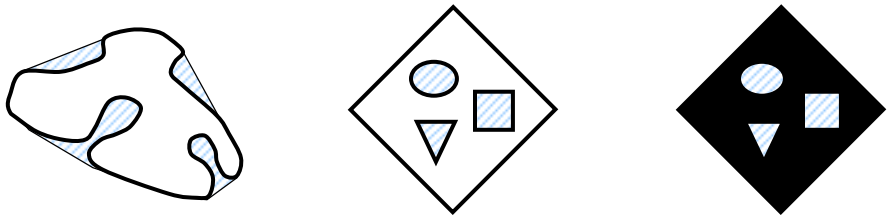
A simplified representation of a curve with two bays is shown in Figure 7.33, omitting several nodes (such as the coordinates of the centers of curvature, the numerosity of the bays, the convex hull of the curve, its barycenter, etc.) to avoid cluttering the drawing. As usual, not all of the available percepts are likely to be perceived and represented in a single run of the image-analysis routines (i.e., before the activation drops to a minimum and activity ceases in the Workspace).

A *circle* is a special kind of curve that Phaeaco recognizes relatively easily, and the only true shape (other than points and lines) that is "hardwired" into the routines of the architecture. This means that there are specific lines of programming code at the retinal level charged with identifying not just circles, but ellipses in general (oriented in any direction). The particulars of the algorithm are

given in §10.3.12. The computational method of representing curves at the retinal level (through piecewise smooth, parametric B-splines) will be explained in §10.3.8.

7.4.6 Concavities and missing area

In §5.1.3, the concept of concave objects was introduced in the context of BP #4 (Figure 5.7), along with the concept of a convex hull. By "subtracting" a filled version of the actual object (if it is not already filled) from its convex hull, Phaeaco is able to perceive the concavities of the object (Figure 7.34a). Besides concavities, the same algorithm discovers other objects inside the original object if it is outlined (Figure 7.34b), or "holes" if it is filled (Figure 7.34c).



(a) (b) (c)

Figure 7.34: Concavities (a), objects inside outlined object (b), and holes in filled object (c)

In addition to identifying such objects and considering them in their own right, Phaeaco is able to obtain an overall perception of "how much is missing" from an object. There is no BP in Bongard's collection based on such a percept, but it is conceivable that a BP could be designed in which the objects on the left side are missing a large portion of their convex hulls (in any of the three ways depicted in Figure 7.34), whereas the objects on the right side are missing considerably less of their convex hulls. Phaeaco represents the percept of "missing quantity" with a single feature node of a continuous nature, depicted in

Figure 7.35. However, this percept is not one of those that are readily available in a first look at the object.

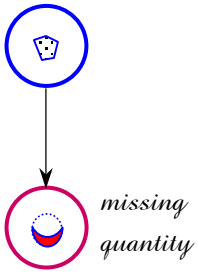


Figure 7.35: Representation of "missing quantity" of an object

In addition to sensing the missing quantity, Phaeaco can also represent explicitly (as separate objects) all the cases shown in Figure 7.34. Concavities of case (a) are imaginary objects (§5.1.3), whereas objects such as those depicted in cases (b) and (c) are real, but "inside" the larger object. The notion of "inside" is the first mention of a relation between objects in the present chapter. This relation will be examined in greater detail in the following subsection.

7.4.7 Interiors

BP #15, a typical problem that distinguishes between objects that have an interior region and those that lack one, was mentioned in chapter 5 (Figure 5.9) as one of the BP's that Phaeaco manages to solve. However, BP #15 appears to be making an "absolutist" distinction between objects that have a completely closed interior and ones in which a narrow "isthmus" connecting the interior to the exterior is sufficient to categorize the objects as "lacking an interior". But, as is the case with most concepts, there can be shades of gray even in a concept as seemingly clear-cut as the existence of an interior, as Figure 7.36 suggests.

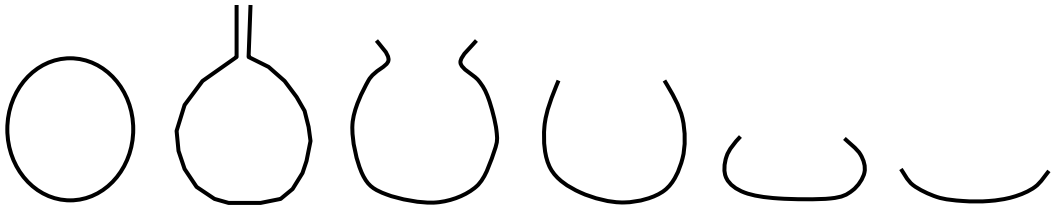


Figure 7.36: Gradations in concept "interior"

Phaeaco detects the "openness" of an interior. For example, on a scale of real numbers from 0 to 1, the leftmost object depicted in Figure 7.36 has an "interior openness value" of 0. As one moves from left to right, the other objects would be assigned larger values, with 1 reserved for a straight line segment. After all, these shapes can be seen as abstractions of real-life objects, such as flasks, phials, vases, pots, bowls, etc., all suitable for storing liquids in their interiors. Phaeaco measures the openness of an interior at the retinal level. First, the suggestion of the existence of such a region is given by the procedure mentioned in the context of concavities (§7.4.6). Once such a region is identified, a few points in it are sampled, as shown in Figure 7.37.

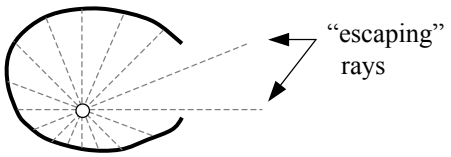


Figure 7.37: Measurement of the openness of an interior region

A sampled random point is shown in the interior of the object in Figure 7.37. A number of "rays" emanate from this point at fixed angles. A few of these rays manage to "escape" to the exterior. By dividing the number of rays that escape by the total number of rays, we obtain a first approximation of how open or closed the object is. By repeatedly sampling a few more points and averaging the results,

Phaeaco makes the approximation reliable. Whereas these computations take place at the retinal level, the cognitive level can only report the magnitude of openness of an interior region qualitatively. Naturally, an object can have more than one interior (e.g., an entirely closed region and a few concavities, an in Figure 7.34a), and the openness of each can be represented as in Figure 7.38.

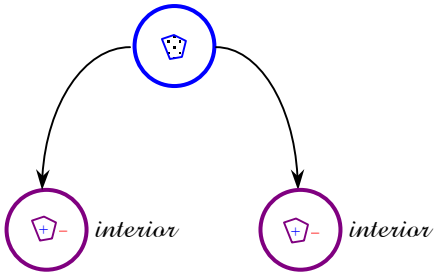


Figure 7.38: Representation of the "openness of interior" of an object with two interiors

It is also important to mention that although the notion "interior" is perceived as a value in a continuous range, Phaeaco can make a sharp distinction between completely closed interiors and all the rest, if the need arises (as in BP #15).

The existence of interior regions entails the possibility of one object being inside another. But just as the notion "interior" admits shades of gray, similarly, the notion "inside" can be blurry, as shown in Figure 7.39.

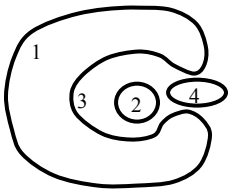


Figure 7.39: The relation "inside" is not always clear-cut

There are four regions labeled 1, 2, 3, and 4 in Figure 7.39. Region 1 is the larger C-shaped outlined object. Region 3 is the "bay", or concavity of region 1

— an imaginary object in Phaeaco's perception. Region 2 is clearly inside the bay of region 1, but region 4 is not easily classified as being inside or outside, and by moving it slightly to the left or right we can vary the degree of "inside-ness". But disregarding for the moment the uncertainty of the inside-ness of region 4, the discussion that follows explains the representations of regions 1, 2, and 3.

First, it is necessary to introduce a new representational structure, one that applies only when relations are present. Suppose we want to represent the fact that region 2 is inside (or "is engulfed by") region 1 (always using Figure 7.39 as a reference point). This is represented as shown in Figure 7.40.

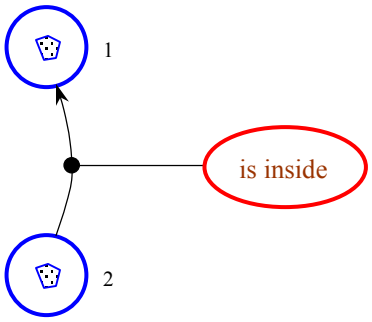


Figure 7.40: Partial representation of relation "object 2 is inside object 1"

There are two object nodes in Figure 7.40, labeled[50] "1" and "2". Object 2 is linked to object 1 by an arrow pointing from 2 to 1, as in previous examples, but this time there is a black dot on this arrow, and a line leads from this dot to the elongated node labeled "is inside". Relations will be depicted with elongated nodes from now on, although they do not differ essentially from other nodes.

Figure 7.40 is strongly reminiscent of drawings of relations in GEB (§6.2, Figure 6.4) and the Slipnet (Figure 6.7). Nonetheless, the similarity is superficial.

[50] As usual, labels are present only for our convenience; in Phaeaco's representations, nodes correspond to objects in the input by virtue of their associated retinal-level information.

Later, when Phaeaco's LTM is explained (in chapter 9), the similarity will be substantial. In Phaeaco's Workspace, however, the arrows that connect nodes in relations are mere "cables": they do not shrink according to the activation of the relational node, which is the most essential characteristic in the Slipnet and in Phaeaco's LTM; instead, they simply denote what is related to what, and how.

The drawing in Figure 7.40 depicts the relation only partially. Every relation is a two-way concept: if one thing is related to another, then the latter is also related to the former, in an inverse type of relation. This is shown in Figure 7.41.

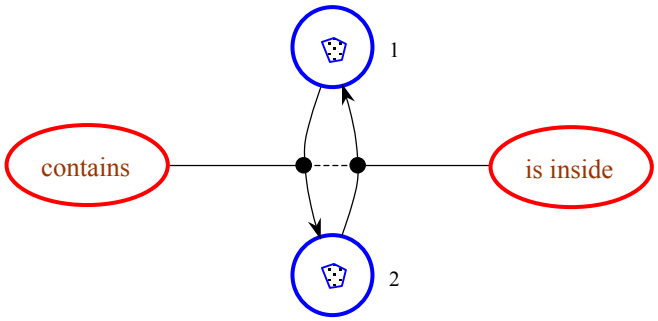


Figure 7.41: Still partial representation of relation "object 2 is inside object 1"

Note the dashed line connecting the two small black dots. Its meaning is that these two relations are not independent; they are like twin siblings. If, for some reason, the relation "is inside" must be destroyed in the future, Phaeaco will "know" what its twin relation is, and will destroy that as well.

The imaginary object 3 (the concavity of object 1) can also be added in the representation. This is not mandatory; it is possible *if* the imaginary concavity has been perceived explicitly. In that case, Figure 7.42 shows how object 3 is related to objects 1 and 2. What cannot be shown in the figure is the difference in degree between these various connections. Specifically, object 2 is *completely* inside the bay (object 3), and so it bears the same degree of "inside-ness" to object 3 as it

does to object 1 (i.e., "completely inside"). However, object 3 is not exactly "completely inside" object 1 (because the two objects touch each other at the right side of their borders), and so the "inside-ness" of object 3 in object 1 is quantitatively different from the previous two cases ("less than completely inside"). The degree of strength of a relation is stored in the structure represented by the small black dot on the arrow that links the nodes, and is implemented as a real value in the range from 0 to 1. For example, a value near 0.5 would represent the inside-ness of object 4 with respect to object 1 in Figure 7.39.

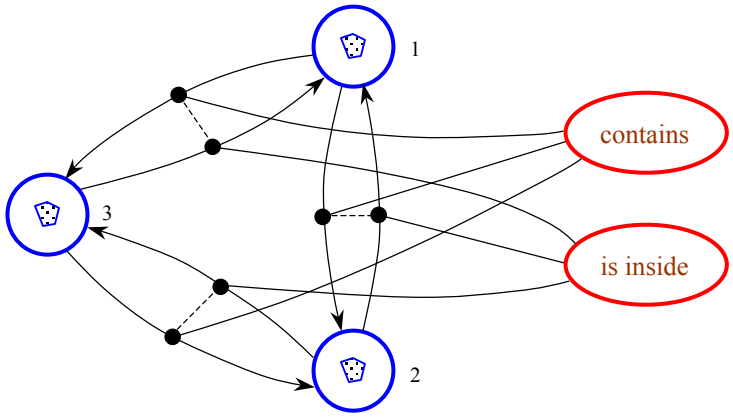


Figure 7.42: Full representation of relations among objects 1, 2 and 3

Naturally, a relation "outside" also exists, with very similar properties to those described above. The only difference is that "outside" is not perceived unless the relation "inside" has already been primed in LTM. This happens as follows. Suppose the input shows an object inside another object. Then nodes for "inside" and "contains" are formed in the Workspace, as already described. These nodes (like all other nodes in Workspace representations) are connected to their corresponding "Platonic" nodes in LTM, with connections that have been omitted in these figures for simplicity. Thus the LTM node "inside" is primed, and

because "inside" is related to "outside" through the concept "opposite", the concept "outside" receives some activation, too. This increases the urgency of Workspace codelets waiting to notice objects outside one another, and when such codelets run, the relation "outside" is explicitly added to the representation.

7.4.8 Elongatedness

At first thought, the elongatedness of an object seems to be one more continuous feature like the ones discussed so far: a straight line is perfectly elongated, and a circle perfectly round. The reality is more complex, however.

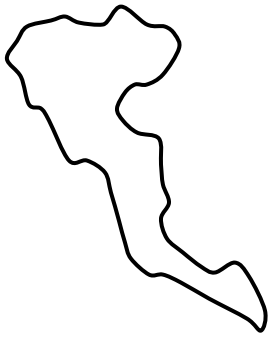


Figure 7.43: What is the value of elongatedness of this shape?

The object in Figure 7.43 does not seem to admit a single numerical value as a description of its elongatedness: it is bulky at the top, narrower at the bottom, and has a varying degree of thickness along what appears to be its length. Even its convex hull seems to suffer from the same indescribability of elongatedness.

Nonetheless, the phrase "what appears to be [the] length [of the object]" was just used, and this notion is not devoid of content. If there is some way to define an internal linear frame, or "skeleton" for this object, then perhaps a value of elongatedness can be assigned along selected points of this skeleton. The

sequence of these values at each point as a whole can be the percept of elongatedness.

Indeed, this is the approach taken in Phaeaco. In the next subsection, the derivation of an internal skeleton for an object will be described. The skeleton, being a "linear structure" (§7.4.5), can be traced by a point along its length, and the elongatedness can be estimated at discrete intervals through an algorithm that is based on another one that examines filled regions (§10.3.6). What is obtained then is a sequence of values (real numbers in [0, 1]) that can be stored in association with the skeleton. Thus, elongatedness is a feature that has a "vector" nature, rather than a scalar one. The elongatedness vector can be given by means of a function $f(x)$, which is defined not by a mathematical formula but by the pairs of coordinates of its points, where x varies discretely in [0, 1], and f yields values also in [0, 1] (where 0 = "low" and 1 = "high" elongatedness).

Reality can be both more complex and more simple than the object in Figure 7.43. For example, the skeleton of an object can be branching, in which case the vector of elongatedness is defined on each branch. But most BP's present us with simpler objects having a linear skeleton and a *constant* elongatedness along that skeleton. In this case, elongatedness reduces to a scalar value. Figure 7.44 shows what the representation looks like in such special cases.

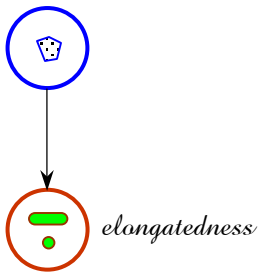


Figure 7.44: Representation of elongatedness in the special case of a scalar value

Many objects in BP's are circles and regular polygons, with an elongatedness of exactly zero (their skeleton is a point). But Phaeaco was not designed with the BP's as its ultimate and limiting domain, so it cannot rely on simplifying assumptions, such as that the elongatedness is either zero or nonzero but constant.

7.4.9 Endoskeleton and exoskeleton

In the biological world, some animals (e.g., the vertebrates) have an internal structure of bones and cartilage called an "endoskeleton"; and others (e.g., the crustaceans) have an external supporting structure, the "exoskeleton". Similarly, one can identify two analogous structures in visual objects. The previous subsection offered a *raison d'être* for the notion of the endoskeleton of an object, an example of which is shown in Figure 7.45.

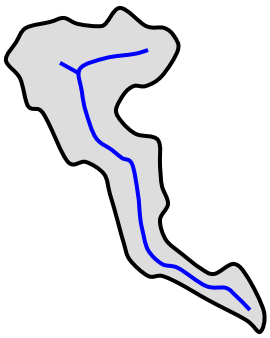


Figure 7.45: The endoskeleton (internal line) and exoskeleton (outline) of an irregular object

The *endoskeleton* is a set of points that stay as far away from the borders of the object as possible. Its computation is deferred to chapter 10, because it is part of the initial stages of visual analysis at the retinal level.

The *exoskeleton* of an outlined object is merely its outline; and if the object is filled, its exoskeleton (also identified at the retinal level) is its set of border pixels.

Of the two structures, the endoskeleton somehow appears to be of more fundamental importance: it is what remains if we stay with the "bare bones" of the object and abstract from it a single line, or a set of branching lines. Although abstracting objects by their endoskeletons is not an operation demanded very frequently in BP's, it is very important in general, given complex shapes.

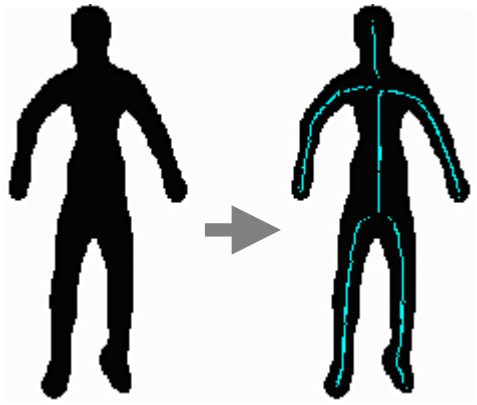


Figure 7.46: A complex figure, and its superimposed endoskeleton as computed by Phaeaco

An example of the utility of finding the endoskeleton is shown in Figure 7.46. Analyzing the structure of the endoskeleton yields useful information about the parts comprising the object. Phaeaco processes the endoskeleton as it would process any other linear structure. Figure 7.47 shows how Phaeaco depicts the fact that an endoskeleton has been computed and incorporated into a representation.

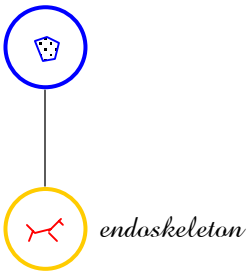


Figure 7.47: Simplified representation of endoskeleton

However, it must be noted that Figure 7.47 has been simplified: ordinarily, an entire network of nodes representing the endoskeleton structure "hangs" under the single endoskeleton node, just as for every linear structure (e.g., a line string).

In contrast, the exoskeleton of a figure does not require an explicit special node for its representation. For example, the exoskeleton of an outlined triangle *is* the lines that comprise the triangle (see the representation of the Λ-shaped object in Figure 7.15 for a similar example); and if the triangle is filled, its representation is the same, except that its "texture" node has value "filled".

7.4.10 Equality, for all

Suppose the input is one of the boxes that belong to the left side of BP #56 (§5.1.4, Figure 5.10), the solution of which is: "all objects have similar texture". A sample box is given below.

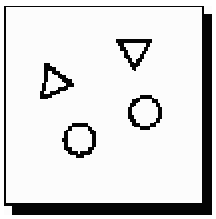


Figure 7.48: One of the boxes of BP #56

A question that must be answered concerns the representation of the phrase "all objects have similar texture". There are two words in this phrase that cannot be represented by any type of node discussed so far: "similar" and "all".

The word "similar" can be used in English in a variety of situations that are not very similar to the way it is used in the above phrase. For example, two election campaigns can be judged as "similar", meaning that people can see the analogous elements among the two campaigns. Or, a model house made of cardboard can be said to be similar to a real house, which need not even exist. In

this case, similarity refers to near-equality that results from bringing the miniature and full-size versions of the two objects to the same scale. In the phrase pertaining to BP #56, the word "similar" refers to equality, or sameness, of textures. Phaeaco can use the notion "similar" in the following cases:

- for absolute equality in values of a discrete feature (e.g., same texture);
- for approximate equality in values of a continuous feature (e.g., same size);
- for approximate equality in numerosity ("same number of"), which reduces to absolute equality for low-numerosity values (e.g., up to about five; see §7.3);
- for similarity of shape, if one object can become identical to another by scale change, rotation, or translation (e.g., a small square and a large diamond);
- for identity of relations (e.g., two or more pairs of objects, each pair consisting of a circle inside a triangle).

If the equality concerns the value of a feature or numerosity (as in the first three cases above), then representations, as they have been presented so far, possess a property that hints at the existence of the equality.

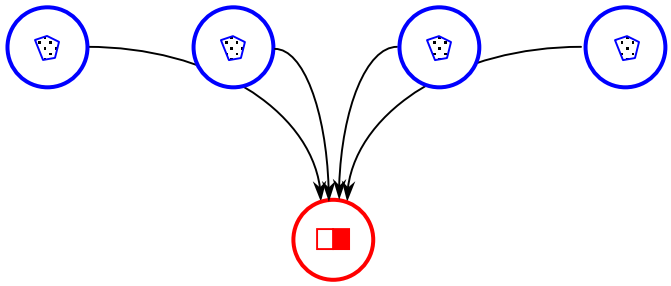


Figure 7.49: Implicit representation of equality (four objects with the same texture)

For example, Figure 7.49 shows four object nodes (representing the four objects of Figure 7.48) sharing the same texture value (presumably "outlined").

That these objects have equal texture can be inferred from the four arrows that point to the same node. Indeed, the texture node stores internally the number of nodes that have been linked to it, so in a sense it "knows" that there are several objects sharing this texture (it maintains an "internal numerosity" value). But this knowledge is implicit, and Phaeaco can also represent it explicitly, as follows.

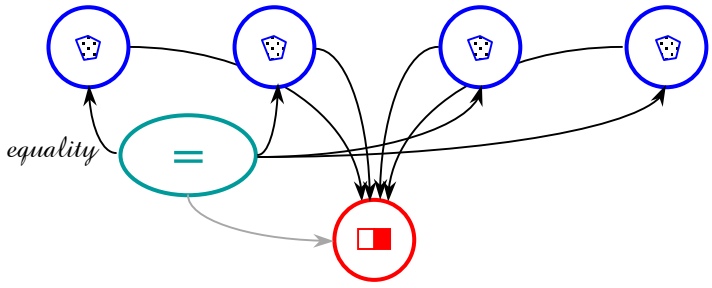


Figure 7.50: Explicit representation of equality of texture value

An oval (relational) node representing equality has been added to Figure 7.50. This node is connected to the four object nodes with one type of link, and to the texture node with a different type of link (the arrow is shown in gray in the figure). A proposition that can be derived from this relation is: "Some objects have outlined texture".[51]

Under what circumstances is an explicit equality node created? Adding a relation in the manner depicted in Figure 7.50 cannot happen during the holistic stage of visual processing (§5.1.1), but during the analytic stage, feature nodes that receive a relatively large number of links increase the probability that a codelet that "lurks" in the Coderack — watching for such structural curiosities — will be selected and run. The higher the number of links, the higher the

[51] This of course is not the only possible proposition. A more elaborate traversal of the linked nodes can derive the proposition "Four objects have outlined texture".

probability for this codelet to be activated. This codelet is analogous to a type of agent referred to as a "sameness detector" ("Sam") in GEB (Hofstadter, 1979, p. 650). After the equality codelet runs and completes its work, it adds a structure as shown in Figure 7.50. What has been omitted from the figure is that the equality node is accessible (linked) from the visual-box node that stands at the "top" of the entire structure.

Equality nodes are a special case of a category of relational nodes called "grouping nodes". Relations such as "is inside" and "is outside" (§7.4.7) also allow the formation of groups by inducing a dichotomy between things inside and things outside. In reality it is hard to find relations that cannot group their referents,[52] so most relations are of the grouping kind.

Nonetheless, the proposition "Some objects have outlined texture" differs from "All objects have similar texture" in two ways: we still do not know how to represent "all" or "generally similar" (with no reference to a specific value).

The concept of "all" follows naturally from what we have seen so far. Once a grouping node is established, it places a high-urgency codelet on the Coderack that wants to examine precisely whether the grouped items are the only ones that can be thus grouped. As was mentioned earlier, the equality node is connected to some "top" inclusive node (the visual box in the example of Figure 7.48). Therefore, if there are no more nodes of the same type (object nodes, in our example) as the grouped ones under the inclusive top node, the codelet concludes that "all" entities that exist have been grouped, and adds the special "for-all" node into the structure, as shown in Figure 7.51. The corresponding proposition that can be derived from this structure is: "All objects have outlined texture".

---

[52] "Successor" and "predecessor" in sequential structures could be non-grouping relations.

Figure 7.51 introduces two new elements. First, a for-all node belongs to yet another type of node, not encountered so far, the "quantifier" (hence the different — hexagonal — shape for its depiction). At present, for-all is the only quantifier implemented in Phaeaco's architecture. (The other important quantifier, "there exists", is implicitly understood every time a representational node exists in a structure.) The second new element in this figure is the LTM node "Platonic object" (depicted with a dashed outline). This is necessary in order to emphasize that the notion "object" in the proposition "all objects have outlined texture" refers not to any particular object among the four ones that have been created in Workspace, but abstractly to objects in the visual box. The *type* of the quantified entity in this proposition is "object", and its *scope* is "those in the visual box".

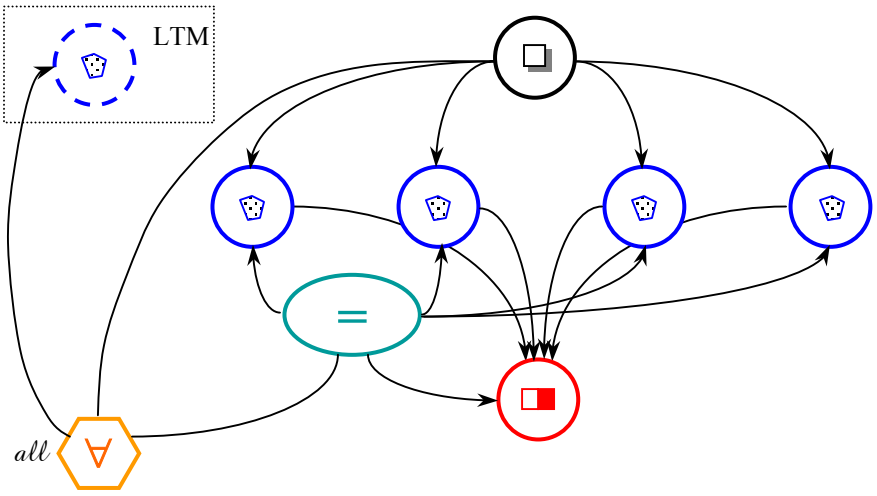


Figure 7.51: Representation of "all objects have outlined texture"

It is interesting to note that there is an obvious analogy between the structure in Figure 7.51 and the logical expression "$\forall x,\ x \in B_o$: *texture*(x) = *outlined*", where B_o is the set of all objects in the visual box. In chapter 4, in the context of the search algorithm RF4, such logical expressions had been singled out as

fundamentally wrong for serving as representations of propositions. Could it be that Phaeaco has been caught in the same *logic* trap as RF4?

Closer examination, however, reveals that the representational similarity is merely superficial. Firstly, Phaeaco is not spoon-fed such propositions by a programmer, but *discovers* them after laboriously examining the visual input, as well as its own representations. Secondly, the node that represents an object in Phaeaco is *grounded* (via a link to retinal-level information) to an object that belongs to the external world, contrary to a symbol such as "*x*"; the same is true for the texture node (grounded to the white color of the input object), contrary to a predicate of the form *texture*(*x*) = *outlined*; and so on. Finally, Phaeaco does not include a calculus to perform logical derivations, or to manipulate formulas and add "theorems" and "corollaries" to the propositions it discovers; it merely "sees" some simple ideas that happen to be describable by what millennia of human inquiry into cognition has distilled into what is known as predicate calculus. Traditionally, it has been asserted that the distilled formulas of logic are fundamental and real, existing in the human brain — perhaps in some module specializing in logical reasoning. In contrast, the approach taken by Phaeaco is that distilled logic formulas exist only in those minds that have been inculcated with the so-called "Western" edifice of scientific knowledge and mathematical reasoning. The edifice is not "wrong" — it is truly beautiful; but the formulaic calculus of logic comprises not its foundations, but the elaborate and fragile furniture of its most recently constructed floors.

However, this supposition does not imply that mathematical logic is the contingent outcome of a random path of human thought, led either by some cultures, such as the Greeks, or by some individuals, such as George Boole, as some authors have recently suggested (e.g., Lakoff and Núñez, 2000, p. 118).

Human thought converged to the formalization of mathematical logic for a good reason: human cognition, together with the rest of our physical makeup, reflects — in an abstract way — the properties of the world in which we evolved. For example, we possess eyes seeing in the visible range of the electromagnetic spectrum because photons that arrive on the surface of our planet belong mostly to that part of the spectrum; we possess lungs to inhale oxygen and blood to circulate it in our body because oxygen, alone among all elements, makes combustion possible; and so on. Likewise, it seems reasonable to assume that we can count objects because objects and numbers exist in the world, rather than that our minds conjure them up; that we can perceive regularities and patterns, such as straight lines, circles, symmetries, and so on, because such regularities are part of our environment, and that our species — occupying the "intelligence niche" — evolved to perceive such regularities because doing so gave us a survival advantage. In contrast, it seems particularly arrogant to claim that mathematics does not exist in the external world, but instead that we humans invented it.

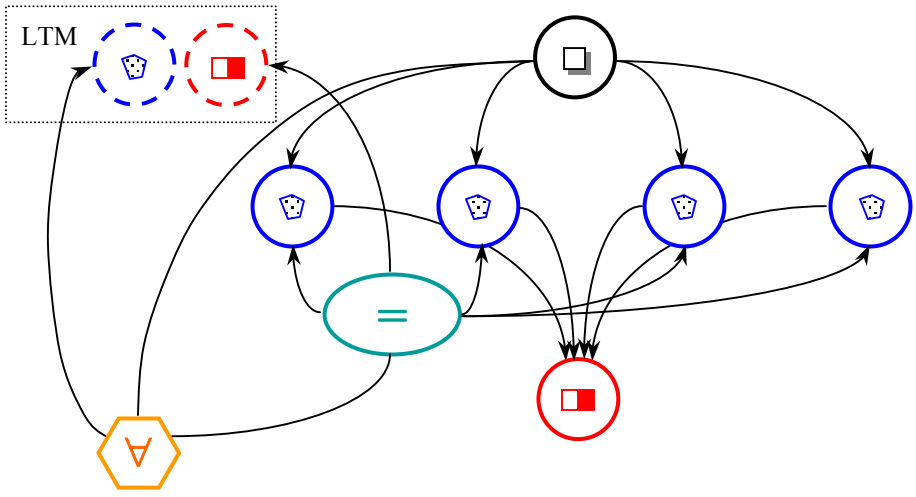


Figure 7.52: Representation of "all objects have similar texture"

Finally, we can now proceed to the representation of the more abstract proposition "all objects have similar texture". The representation is shown in Figure 7.52, and involves one extra step of abstraction: the arrow that previously connected the node of equality with the outlined texture node in Workspace now points to the Platonic node "texture" in LTM. All else is identical to the structure in Figure 7.51.

7.4.11 Necker views

A *Necker cube* is a well-known object in psychology, first noted in 1832 by Swiss crystallographer Louis Albert Necker. Our perception of its structure constantly flips back and forth between the two possible ways to perceive the three-dimensional arrangement of its sides and edges (Figure 7.53).

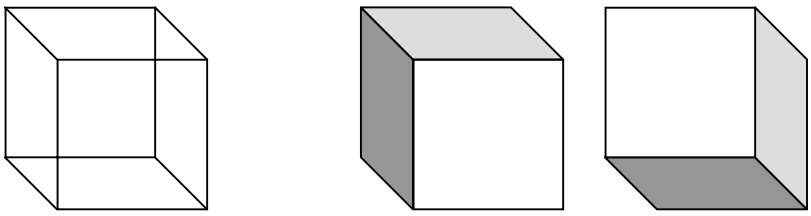


Figure 7.53: Necker cube (a), and two possible ways to perceive it in three dimensions (b)

What is interesting about the Necker cube is that we feel there is no way to entertain *simultaneously* its two "conflicting" representations in our minds: we may focus on one view or the other, but not on both at the same time.

The Necker cube situation appears often in Phaeaco's input, even in drawings not nearly as complex as a Necker cube. Figure 7.54, for example, shows a very simple yet ambiguous drawing: is it a short line segment, or a very narrow filled rectangle? How is such an object to be represented? Should one representation

(e.g., "straight line segment") be favored over the other? Should the disfavored representation be forgotten and deleted from Workspace? But what if the context changes, necessitating the "other" representation? Is the forgotten representation to be constructed once again, as if the system has never seen it before, forgetting the first (original) representation this time?

Figure 7.54: A simple ambiguous drawing: line segment or filled rectangle?

Phaeaco answers this dilemma by keeping the two representations in parallel (assuming it reaches both at some point) by means of a special representational element: a "Necker view node", shown in Figure 7.55.

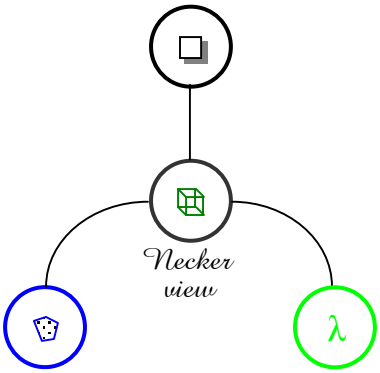


Figure 7.55: A Necker view representation

The important function of a Necker view node in a representation is that whenever an operation must be performed on the representation, one of the two links connected to the node must be followed, hence one of the two views must be assumed. (Which view will be assumed is a matter of contextual pressures.) For example, the matching and statistical updating of representations is introduced in

the next chapter; if the matched or updated representation contains a Necker view node, then the algorithm that performs the matching, updating, etc., will be forced to "choose" one of the two views in order to proceed.

Any number of views can coexist under a Necker view node, which is one of the possible ways to represent the idea of "or", or "alternatives". In a way, the Necker view node acts as if it were not present, because after one of its views has been assumed, the alternatives become temporarily inaccessible. Nonetheless, the system is aware that alternative views exist, so that given a suitable contextual pressure it can switch to one of the alternative views without reconstructing it from primitives.

7.5 Some general remarks on visual primitives

Confronted with the BP domain, either as a solver or a designer of problems, one at first develops the impression that the variety of possible ideas that can be expressed in BP's is endless — quite literally *infinite*. Another possibility is that the set of available BP's, though not literally infinite, is so vast that any attempt to describe it thoroughly with a collection of examples is destined to fail. However, it was mentioned in an earlier chapter (near the end of §5.1.4) that a set of approximately 10,000 different BP's could include the ideas of all but the most creative designers. Which view is right?

The notion of a true infinity of BP's can be easily shown to be false, assuming the discrete nature of the input.[53] Suppose each of the k boxes of a problem has dimensions $m \times n$ (for example, usually $k = 12$ in BP's, and $m = n = 100$ in

[53] "Continuous" input can be discretized to any desired degree of accuracy, hence the argument presented here includes continuity as its limiting case; in addition, it is doubtful that continuity exists in the physical world in any way other than as an idea in human minds.

Phaeaco's input). Then there is a maximum of $2^{k \cdot m \cdot n}$ possible black-and-white "BP's" that can exist, and that is the end of the infinity hypothesis.

Still, using Phaeaco's default assignment to *k, m,* and *n,* the number $2^{k \cdot m \cdot n}$ (i.e., $2^{120,000}$) is vastly larger than the total number of elementary particles in the observable universe. If we wanted to keep a record of all possible BP's, the entire known universe would not suffice. Could there be any validity to the notion that the number of *interesting* BP's is so large that it cannot be adequately approximated with any manageable collection of examples?

There are two ways in which a designer can come up with interesting new BP's: by discovering new visual primitives or by defining new concepts using known primitives.

The second method indeed leads to a large collection of BP's, but inevitably, an observer of the collection will find most such problems esoteric. For example, given the concept of a triangle, one may define a multitude of "centers" associated with it; some of them are well known (e.g., barycenter, incenter, circumcenter, orthocenter); but other centers are little known to the non-expert (excenters, nine-point center, Nagel point, symmedian point, Fermat point, Gergonne point, Morley centers, Hofstadter one and zero points, and many more). One may then consider lines that pass through such, often collinear centers (e.g., the Euler line), circles defined by them, their points of intersections that define further triangles, and so on. In this way, an entire "geometry of the triangle" has been developed (e.g., Kimberling, 1998). But now consider a BP that includes a single triangle in each of its 12 boxes, the solution of which is as follows: on the left, the Euler line of each triangle is vertical; whereas on the right, the Euler line is horizontal. This BP would be completely unapproachable by the vast majority of people; and even the few experts who have dedicated entire decades to the study of triangle

geometry would need to be given some hint that this BP is relevant to their area of expertise before they could adopt the necessary approach.

The Euler-line BP that was just proposed is an extreme example, but it helps to appreciate the idea that the method of making definitions of exotic concepts does not result in universally acceptable BP's. What remains as a possibility is the discovery of new primitives, which brings us to the question of approximately how many primitives exist. It should be noted that some notions counted as "primitives" in this chapter are not primitive in a mathematical, "axiomatic" sense. For instance, the midpoint of a line segment could be defined on the basis of its two endpoints, and the equality of lengths of two subsegments of the given segment. But it is "primitive enough", in the sense that when people perceive it they do not think of it as definable in terms of other notions they already know. All such notions are considered "primitive" in the present text, even if they entail some redundancy.

A supposition in this thesis is that the set of visual primitives that can be expressed in BP's is large — probably of the order of a few hundred. The magnitude of this set generates the feeling that the possibilities in the domain are endless. Since each BP usually does not depict exactly one primitive but is an elaboration, or a combination of a few, the number of BP's that are original enough to surprise the solver is, perhaps, of the order of a few thousand. But beyond some point there is bound to be repetition, recycling, and recombination of old ideas, once the set of all primitives is exhausted.

That the set of primitives is large explains why no attempt is made in the present thesis to list them exhaustively. For many entries on such a list, the decision of whether they constitute primitives or not would be subjective. In addition, the magnitude of this set also suggests that an attempt to implement its

members exhaustively would be the wrong approach. Instead, only a few representative members of each category of primitives (features, objects, relations, and quantifiers) have been selected and implemented in the current version of Phaeaco.

7.6 Summary

The principles by which Phaeaco constructs representations were introduced in this chapter. Representations consist of nodes linked to each other in a structured, nearly hierarchical manner that reflects the corresponding hierarchy of levels of detail of the represented object. Some representational characteristics, such as the notion of activation, as well as some representational primitives, such as the numerosity of objects, were discussed in detail. Other primitives were briefly introduced, broadly categorized as objects, features, relations, and quantifiers. But just as data structures are not very attractive in computer programming unless supplied with operations that act on them, so representational structures are not very useful in a cognitive architecture unless accompanied by a "calculus" that operates on them in some useful way. The next chapter addresses this issue.

CHAPTER EIGHT

Visual Patterns

The representations introduced in the previous chapter become "functional" in the present one. First, the motivational problem of forming "lumps" (groups) of similar items within a population is presented. Next, a general matching method is offered that, given any two items represented as discussed in chapter 7, returns a real number that stands for the "perceptual difference" between the items. This difference is then used as a metric to solve the problem of "lumping" by assigning each item to some group, or category. The number and identity of the formed categories depends on contextual pressures. Finally, the formation of a visual pattern in each such category is discussed.

8.1 Motivation

Consider the three examples of visual input given in Figure 8.1.

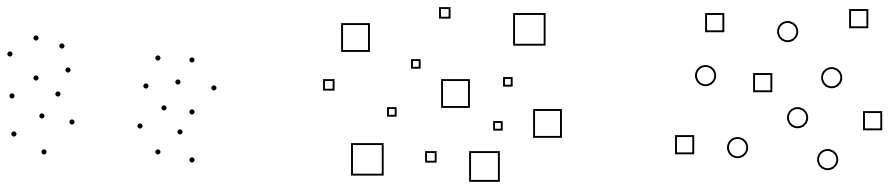


Figure 8.1: Three examples of "lumping": by location (a), by feature value (b), and by shape (c)

When presented with one of the examples shown in Figure 8.1, we can place the objects into groups by "lumping" them together. In some cases this lumping is purely automatic and unavoidable, as in Figure 8.1a, in which the locations of the objects (dots) cause us to see two groups. In other cases, a minimum of effort is required, as in Figure 8.1b, where two groups are formed according to the size of the objects (large and small squares). Interestingly, the ease of group separation can be modulated by varying the "statistical distance" between the two samples, which depends both on the absolute difference of the two means (mean area of large and small squares), and on the variance of each sample (the larger the variance, the harder it is to identify the groups). Finally, Figure 8.1c gives an example of group formation based on the shape of the objects (squares and circles). In this case it is the structural properties of the given items that matter.

The problem of group formation appears to be fundamental in perception. In §8.4 it will be argued that this is the most essential problem cognitive agents must solve before they can "understand" anything at all in their environment. Only by solving this problem we can perceive "objects" in the world (more in §8.4; for now, note that we see "two swarms of bees" in Figure 8.1a, rather than "bees" in random locations). At the core of the solution of this problem must be some procedure that allows us to compare the given items and determine how different, or "distant" (either literally or abstractly) they are. The procedure, called "pattern-matching" in the present text, is part of a more elaborate algorithm that allows us to form both mental categories of items and a summary representation for each category: a "visual pattern" (§8.3). Once a visual pattern is formed, we are able to determine with some confidence how plausible it is that a given new item belongs to the category. It will be argued at the end of this chapter that pattern-matching

stands at the foundation of a more general mechanism that Hofstadter calls "analogy-making".

How can a system like Phaeaco, which exists in the realm of "programmed cognition" (§4.3, see also Figure 4.9), achieve the "same thing" as we do, i.e., lump items into groups? The rest of this chapter explains Phaeaco's approach.

8.2 Pattern-matching

The algorithm presented in this section accepts any two representations (as described in chapter 7), and outputs a real number in the interval [0, 1] that indicates how well the two representations "match", i.e., how similar they are. It is now appropriate to use the term "exemplar" to refer to the structures of chapter 7, and to reserve the term "representation" for something that is either an exemplar or a "pattern". The latter term is introduced in §8.3 to refer to representations that are statistical summaries of exemplars. Thus, "representation matching" would be a more suitable term for the main algorithm of the present chapter, because its inputs are representations in general (either exemplars or patterns). However, the term "pattern-matching" is easier to understand because it has been used extensively in the literature, so it will be used instead.

An issue that must be addressed before the algorithm is presented is what triggers the algorithm to run. When are two representations matched against each other? The answer is that the algorithm is activated probabilistically when two or more representations exist under the same context; and the larger the number of items, the higher the probability that the algorithm will run. For example, a BP side contains six boxes, which is a number large enough to result in a very high probability that the algorithm will run with pairs of representations of boxes as

input.[54] Also, any of the examples (a)–(c) of Figure 8.1 will almost certainly cause the algorithm to run. Even if there are only two items in a group, they can be matched, but in the absence of any pressure there is a nonzero probability that this might not happen. The probabilistic start of the algorithm is implemented with a codelet that monitors the appearance of a high numerosity of "objects" — the same codelet that was mentioned in the context of the for-all quantifier (§7.4.10).

The description of the algorithm that follows proceeds in a bottom-up fashion: it begins with feature nodes, and later continues with entire structures of the two matched representations.

8.2.1 Matching feature nodes

Suppose two feature nodes f_1 and f_2 of the same type (excluding numerosity) are given. Suppose further that the field "N" of each of f_1 and f_2 (described as "number of observations" in Table 7.1) has value 1 (i.e., each of the two features has been seen only once, as is the case of any exemplar representation built from a single observation, as described in chapter 7). Then the distance between f_1 and f_2 is the absolute value of the difference between their corresponding fields labeled as "mean" (also described as "average value of sample data" in Table 7.1), divided by the maximum value a feature of this type can acquire. The last operation has a scaling effect, so that the distance is always a number in [0, 1]. For example, if the feature type is "length", the maximum value is the length of the main diagonal of the visual box; if the feature type is "angle", the maximum value is 180°; if it is "slope", the maximum value is 100%; in general, each primitive feature type "knows" what its maximum allowed value is. The result is

[54] Note that in this case Phaeaco does not rely on probabilities, "knowing" that the representations of the six boxes must be matched; in other words, the algorithm runs with probability 1, because Phaeaco "knows" what a Bongard problem is and what to do with it.

summarized in Equation 8.1, where $\bar{x}_1$ and $\bar{x}_2$ are the values of the field "mean" of f_1 and f_2, respectively, and $\max(f)$ is the maximum possible value of feature f.

$$d = \frac{|\bar{x}_1 - \bar{x}_2|}{\max(f)}$$

Equation 8.1: Distance d of two features with sample size 1 (two exemplars)

If the field "N" of feature f_1 is equal to 1, but the field "N" of feature f_2 is greater than 1 (i.e., f_2 is part of a *pattern,* to be explained in §8.3), then the distance d between the two features is equal to the probability that f_1 comes from a different population than the one described by the statistical sample of f_2. Computing this probability is elementary statistics. Phaeaco uses the Student's t distribution (Equation 8.2).

$$d = \frac{\int_{-t}^{t} \left(\frac{N-1}{N-1+x^2} \right)^{N/2} dx}{\sqrt{N-1} \int_0^{\pi/2} \sin^N x \, dx}, \quad \text{where} \quad t = \frac{|\bar{x}_2 - \bar{x}_1|}{s/\sqrt{N}}$$

Equation 8.2: Distance d of a feature f_2 of sample size N from a feature f_1 of an exemplar

In Equation 8.2, $\bar{x}_1$ is the field "mean" of f_1, and $\bar{x}_2$ is the "mean" of f_2, which is of sample size N and deviation s. The integral[55] in the numerator returns the area under the curve of Student's t distribution, and the integral in the denominator is a constant depending on N such that the total area under the curve is 1.

Finally, suppose the fields "N" of both features f_1 and f_2 are greater than 1 (i.e., both f_1 and f_2 are parts of corresponding patterns — see §8.3). Then the

[55] Gauss–Legendre numerical integration is used to evaluate integrals in constant time.

distance d between f_1 and f_2 is equal to the probability that the populations described by the statistical samples of f_1 and f_2 are different. To express this in formulas, assume f_1 has mean value $\bar{x}_1$, standard deviation s_1, and sample size N_1; and f_2 has mean value $\bar{x}_2$, standard deviation s_2, and sample size N_2. Then the desired probability is found by substituting in Equation 8.2 the values of N and t given in Equation 8.3.

$$N = \left(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2} \right) \Bigg/ \left(\frac{\left(s_1^2/N_1\right)^2}{N_1 - 1} + \frac{\left(s_2^2/N_2\right)^2}{N_2 - 1} \right) \qquad t = \left| \bar{x}_1 - \bar{x}_2 \right| \Bigg/ \sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}$$

Equation 8.3: Values to substitute for N and t in Equation 8.2 for the distance between two features f_1 and f_2, with statistics $\bar{x}_1$, s_1, N_1, and $\bar{x}_2$, s_2, N_2, respectively

N in Equation 8.3 is the "degrees of freedom" of the Student's t distribution, and t is a "standardized t-statistic".

8.2.2 Matching numerosity nodes

The equations given in §8.2.1 concern feature nodes that are not numerosity nodes. If the nodes are of type "numerosity", then the formulas are different, but the general idea is the same as in §8.2.1. Specifically, a distance is computed again by comparing samples from Gaussian-like populations. Recall that the representation of a number in Phaeaco, according to the accumulator metaphor (§7.3.2), is a Gaussian, given in Equation 7.3. The method of comparing two numbers was given as an algorithm in §7.3.2, where it was noted that, if the two numbers to be compared are L and S (for "Large" and "Small"), then their standard deviations are $\sigma_L = \sigma_0 \sqrt{L}$ and $\sigma_S = \sigma_0 \sqrt{S}$, respectively, where σ_0 is some constant. Then their distance d is the probability that the populations described by the Gaussians N(L, σ_L), and N(S, σ_S) are indeed different. Suppose the two values

L and *S* have been observed only once (i.e., their field "N" in Table 7.1 is 1). Then *d* is computed by the following formula.

$$d = \frac{1}{\sqrt{2\pi}} \int_{-z}^{z} e^{-\frac{x^2}{2}} dx \text{ , where: } z = \frac{|L - S|}{\sigma_0 \sqrt{L + S}}$$

Equation 8.4: Distance *d* given two numerosity values *L* and *S* that have been observed once

Equation 8.4 is analogous to Equation 8.1: it tells us how to compute the distance between two numerosity values that have been observed once. Formulas analogous to equations 8.2 and 8.3 for numerosity values with a statistical sample of size greater than 1 are given below. To avoid any possible confusion, note that the notion of numerosity is associated with the idea of a "statistic" in two different ways: implicitly, through the accumulator metaphor and the resulting Gaussian spread of the perceived numerosity value around the "correct" number; and explicitly, because each numerosity node is also of type "feature" in Phaeaco's ontology, so that it includes the fields of Table 7.1. Examples of how the statistics of numerosity can be collected explicitly would be to let the system perceive a sequence of groups of dots (similar to those in Figure 8.1a), each group consisting of a different number of dots; or to be exposed to a sequence of different polygons — heptagons, octagons, nonagons, etc. — noticing the numerosity of their sides, or vertices. In such cases the field "N" ("number of occurrences") of the numerosity value is greater than 1, and all other fields ("mean", "var", etc.) in Table 7.1 are updated.

There is a formula analogous to Equation 8.2 for computing the distance between two numerosity nodes, one of which has been observed only once (hence its accumulator-generated Gaussian has mean *M* and standard deviation $\sigma_0 \sqrt{M}$), and the other is a statistical sample of size $N_2 > 1$. Recall that the sum of variables

from a Gaussian distribution is again a Gaussian (§7.3.2), so denote the mean of the second numerosity node by $\bar{x}_2$ and its standard deviation by s_2. The formula is similar to Equation 8.4, except that the standardized variable z is as follows:

$$z = \frac{\left|M - \bar{x}_2\right|}{\sqrt{\sigma_0^2 M + \frac{s_2^2}{N_2}}}$$

Equation 8.5: Value of z to use in Equation 8.4 for numerosity values one of which is observed once (an exemplar), and the other $N_2 > 1$ times (a pattern)

Similarly, the formula for numerosity nodes both of which are samples of size greater than 1 (i.e., two patterns, analogous to Equation 8.3) is similar to Equation 8.4, except that the standardized variable z is given by the following formula:

$$z = \frac{\left|\bar{x}_1 - \bar{x}_2\right|}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$

Equation 8.6: Value of z to use in Equation 8.4 for numerosity values of two patterns

The parameters in Equation 8.6 ($\bar{x}_1$, $\bar{x}_2$, etc.) are assumed to have their obvious meanings.

8.2.3 Combining feature differences; contextual effects

The formulas in §8.2.1 and §8.2.2 indicate how to compute the difference between two feature nodes in isolation (e.g., two slopes), but they do not explain how to combine the features shared by an object. For example, suppose that two straight line segments are compared, hence their representing λ-nodes must be matched with each other. Each line segment has several features: slope, length, etc. Given the difference d_1 between the two slopes, and also the difference d_2

between the two lengths, and so on, how do we combine these differences to arrive at a single value d that will denote the overall difference between the two line segments? Figure 8.2 depicts such an example.[56]

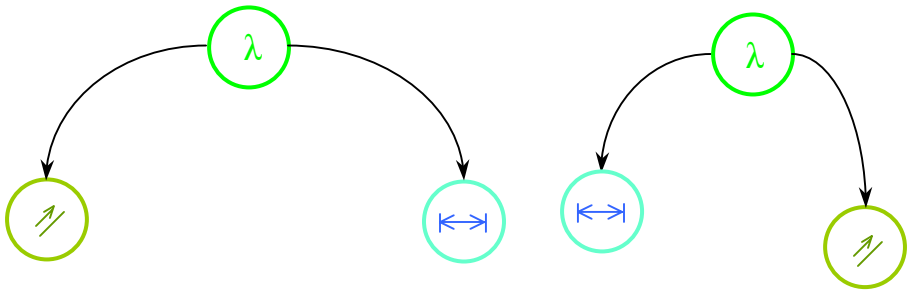


Figure 8.2: Two line segments with their features considered for matching

In general, suppose there are n features in each of the two compared objects, and let d_i, $i = 1,\ldots,n$ be the differences of their corresponding features, where each d_i is computed as explained in §§8.2.1–8.2.2. Then the overall difference d of the two objects is computed by Equation 8.7.

$$d = \frac{\sum_{i=1}^{n} w_i \, d_i}{\sum_{i=1}^{n} w_i}$$

Equation 8.7: Overall difference of two objects with a number of features

The w_i's are the weights of the compared features. Each w_i is a non-negative real number, and equals the current "significance" of the corresponding feature in LTM. For example, suppose the d_i between two slopes is computed. The node

[56] The numerosity node is not shown in Figure 8.2 because the algorithm matches numerosity values at a prior stage; for example, the difference in numerosity of line segments that are part of an object is computed at the stage when the two object nodes are matched, which occurs prior to matching the λ-nodes of the line segments.

"Platonic slope" in LTM has a significance value w_i that, as will be explained in chapter 9, consists of two parts: a relatively permanent long-term significance (that represents how important the abstract notion "slope" is for the system), and a relatively temporary activation value (that corresponds to how much the notion "slope" has been primed at the current moment). Both the permanent significance and the temporary activation are added and result in a value for w_i. Given that each d_i is a number in [0, 1], it follows easily from Equation 8.7 that d is also a number in [0, 1].

If the i-th feature is missing from one of the two representations (e.g., because the corresponding codelet was not given a chance to run and add the feature in the representation), then the value of d_i is assumed to be 0 (i.e., the feature is ignored).

The role of the w_i's is fundamental in how the context determines the notion of psychological distance. Recall that each w_i is not restricted to [0, 1], but can take on any non-negative value. Thus, if one of the features is highly primed in LTM, its high w_i (relative to the rest) can cause the comparison to be made essentially solely on the basis of this feature, all other features contributing a negligible amount in the calculation of d. For example, consider the display in Figure 8.3, in which there are several polygons with a variety of features: they can be differentiated according to their number of sides, texture (outlined or filled), size, line width, or any other of their features. Suppose that Phaeaco "wants" to concentrate on the notion "texture", and that, by doing so, the activation of its Platonic idea "texture" in LTM is increased so much that its significance (w_i) dwarfs all other possible features. Then two objects with the same texture would be judged as practically "the same" ($d \approx 0$), whereas two objects with different textures would appear as different as two objects can be ($d \approx 1$). In this way the

stage is set for two groups to be formed, one with filled and another with outlined objects. The mechanics of how to use *d* to form the groups is explained in §8.3.

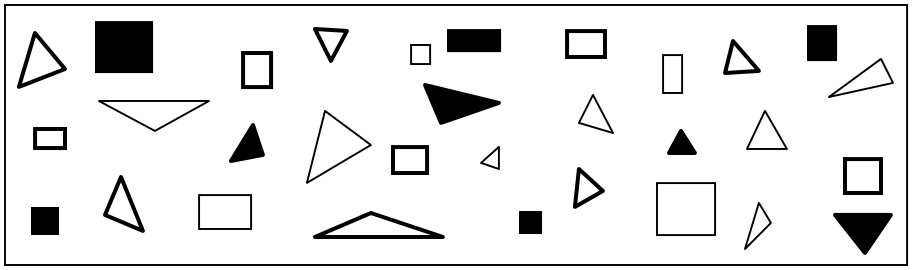


Figure 8.3: Objects with a variety of features

Similarly, by shifting the focus of attention to different features and activating their Platonic representations, other group formations are possible from the same input. Note that the described contextual group-formation does not attempt to model the human ability to see, e.g., the filled objects "popping out", which is probably related to lower-level, retinal processing properties (Treisman, 1980).

8.2.4 Matching entire structures

The formulas in §§8.2.1–8.2.3 correspond to the "base case" of a recursive procedure. When the representations to be matched do not consist of a single node of type "feature" but of an entire structure (such as those depicted in several of the figures in chapter 7), then the recursive part of the algorithm is invoked. Nonetheless, the word "recursive" at Phaeaco's cognitive level is not identical in meaning with the same word as used in mathematics and computer science. An example of a high-level analogy will clarify the difference.

Suppose an analogy is being made between the governments of two nations, the U.S.A. and Germany. The American president might be mapped onto the German president, but given that the former is the chief executive of the

government whereas the latter's role is head of state rather than head of government, a mapping between the American president and the German chancellor might appear more appropriate. Next, one would have to select the best matches for the American vice president and secretary of state, the German vice chancellor, who also acts as the chief executive for Foreign Affairs, the American House of Representatives and the Senate, the German Bundestag and Bundesrat, the roles of various ministers, and so on. Generally, the analogy would be far from perfect, and different mappings might be considered in an attempt to find the best overall match. Though the final matching value would be subjective, most people would feel comfortable with and agree on some mappings, while disagreeing on others.

The analogy is usually constructed by proceeding in a top-down fashion: we feel we have to start with the most "important" government members and structures first (their importance being understood according to the impact their decisions have on people's lives), and then proceed to less important people or committees, *if at all*. The last qualification is important: we do not proceed blindly, mapping the two ministers of agriculture to each other if we have failed to find a good match at the level of president and chancellor. Also, we proceed with mappings at less important positions ("recursively") only up to some point, beyond which any further mappings apparently cannot have a significant effect on our overall "feeling" of how good the match is. Finally, although each particular person in a government possesses a substructure consisting of many levels (head, body, arms, and legs, each being a structure in itself), we feel it is absurd to match, for example, the American president's nose with the German chancellor's nose; or to entertain for a moment a match between the nose of a person and the left eye of another, and upon failing, to consider next the right eye, and so on.

Yet, in spite of the absurdity of this suggestion, there exist computer models of analogy-making that employ precisely such mechanisms of blind mappings and recursive searches to find the best possible match between two structures, such as ACME (Holyoak and Thagard, 1995), and SME (Gentner, 1983). Some of these models have even been designed with a neural network architecture, as if the fundamental problem of answering how an analogy is identified has been solved, and so all that remains is to show how neurons can come onto the stage and effect an *implementation* of the theory, "proving" that analogy-making, after all, is possible in agents that compute with neurons. (For a review of the pitfalls of such approaches, see Hofstadter, 1995b.)

Phaeaco, in contrast, although engaging in the humbler pursuit of visual pattern-matching, places some limits on the degree of recursion allowed by its algorithm. For example, why does matching noses and ears of presidents seem so utterly irrelevant for the purpose of deciding the structural goodness of a match between two governments? The answer is that in Phaeaco's approach bodily parts belong to a different level of organization, and the mixing of levels is what imparts absurdity to such matching attempts. The notion of "different level" in this example arises from the observation that the idea "human being, person" forms a well-understood category; therefore, when two people are given, we *already* know they "match" with each other as members of the same category, so we do not need to move "down" into the category level, and examine how well two people match structurally. The move from the category "government" to the category "person" implies a change of similarity criteria (from "importance and role in government", to "physical similarity of bodily parts"), and this is the main reason why the mixing of levels appears incongruous.

Another mechanism that prevents deep recursion in Phaeaco is, as was hinted above, an application of the notion of "diminishing returns". Representations, as introduced in chapter 7, possess to a large extent a hierarchical structure. For example, a visual box can contain some groups of objects, with each group made of individual objects, each object made of lines, vertices, curves, etc. Although the structure is not a tree but a graph, it closely approximates a tree. Thus, the pattern-matching algorithm starts at what appears as the "root" of this tree-like structure (for example, it could be the node of the visual box, if a match is attempted between two boxes of a BP), and proceeds to the subparts of the structure. Each "downward" step along nodes that are accessible from the top node increases substantially a quantity that registers the depth the process has reached, taking into account the relative importance of the visited nodes. Two or three recursive steps "down" are usually enough to discourage the algorithm from proceeding any further into the structure. Thus, "deep recursion" is ruled out in Phaeaco in more than one way.

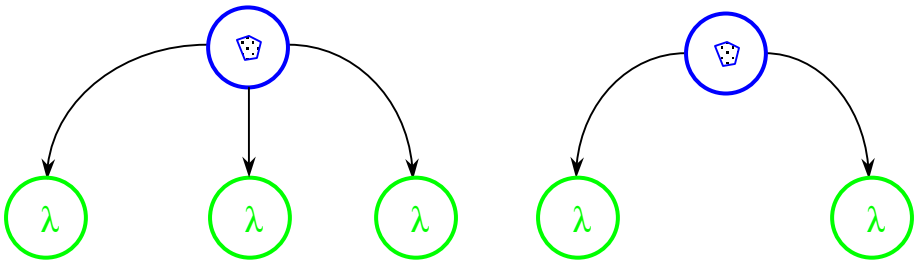


Figure 8.4: Two structures to be considered "recursively" for mapping

Figure 8.4 shows two sample structures to be matched with each other. Knowing how to find the difference between two line segments, for example (as explained in §8.2.3), and applying a "recursive" descent — with the qualifications of the present subsection — Phaeaco finds an overall difference for the two object

nodes shown in the figure. Observe that in finding the best match for a λ-node on the left-hand structure, Phaeaco will compute its difference with each λ-node on the right-hand structure before choosing the best match; then it will proceed to the next λ-node on the left, and so on. Even this seemingly "computeristic" point of the algorithm, however, is only of order O($n \cdot m$), where n and m are the numbers of compared nodes of the left and right structures — which is not comparable to an exponential explosion.

After discovering the best matches, the algorithm can also record which elements of one structure correspond to which elements of another. This will be useful later, when Phaeaco is building the average pattern of a category (§8.3.3).

8.2.5 Using difference to compute similarity

Once a value of difference d is computed as explained in the previous subsections, a measure of similarity s can be obtained via the following transformation:

$$s = e^{-d}$$

Equation 8.8: similarity as a function of difference

Since the value of d in Equation 8.8 is in the range [0, 1], it follows that the value of s is in the range [1/e, 1], where 1 stands for "identity", and $1/e \approx 0.37$ for "minimum similarity". Although a value such as 1/e appears to destroy the elegance of the range for differences, [0, 1], note that any choice of range is, after all, arbitrary. Future extensions of the architecture beyond the BP domain might require extending these limits as well (e.g., to "even less similarity" than 1/e). Equation 8.8 is employed merely for compliance with the Generalized Context Model (§6.1.4).

8.3 Group and pattern formation

Equipped with a similarity measure for representations, we can now proceed to solve the fundamental problems of identifying groups of objects, and forming a summary representation (a "visual pattern" in Phaeaco's terminology, or "pattern" for simplicity) for each group. It is best to do this by considering a concrete example: the quintessential group-identification BP, already presented in §0 (Figure 2.9, BP #166), a single box of which is shown in Figure 8.5.

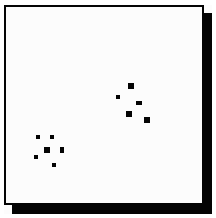


Figure 8.5: One of the boxes of BP #166, exemplifying group formation

It is desirable to have an algorithm that, given the input box shown in Figure 8.5, discovers two groups of objects (dots), and builds them incrementally, as the dots are "seen" one by one (i.e., as their representations are built in Workspace one by one). This is an absolute requirement: in Phaeaco's fashion of processing input, it is not possible to wait until all objects become available before manipulating them, because it is not known when an "end of input" will be detected. For this reason, all processing algorithms in Phaeaco start working as soon as the first data on which they can work become available. Nonetheless, this requirement — which is dictated to Phaeaco by its architectural constraints — does not imply that human cognition, too, always groups entities sequentially and incrementally. The dots in Figure 8.5, for example, are probably processed in parallel by the hardware of our retinas and visual cortex, whereas the process of

learning the stereotypical behavior of people belonging to a particular culture might take a person years or even decades, so it can be reasonably assumed to be achieved incrementally, without having immediate access to the entire set of data. This observation, as we shall see in the subsections that follow, permits some freedom concerning the kinds of algorithms that can be assumed "admissible" (in a sense that will be explained shortly) for a cognitive agent.

8.3.1 Background from computer science

The requirement that the group-formation algorithm be incremental immediately disqualifies a number of methods used in computer science in the field of data mining, in which algorithms often build groups after all data have been collected, and the number of desired groups is given as a parameter (for a review, see Jain, Murty *et al.*, 1999). The notion of unsupervised group-formation in data mining is usually called "clustering". Some non-incremental or supervised algorithms (e.g., "*k*-nearest neighbors") can perform optimally because they benefit from the re-examination of the input, but they are hard to reconcile with cognitive mechanisms, which clearly do not store data in large arrays, scan them multiple times, and store intermediate results in other data structures, nor do they receive supervision from an "oracle" that knows the classification of a subset of the data.

An additional requirement that disqualifies a large class of algorithms is that there can be no parameter such as "desired number of groups" that is known in advance; clearly, in cognitive tasks the number of groups is one of the features that must be identified. Thus, clustering algorithms such as "*k*-means" (McQueen, 1967) and its variants are cognitively implausible. Despite its implausibility, however, the simplicity of *k*-means makes it useful as one of the standards against which Phaeaco's algorithm is compared in §8.3.4.

Incremental algorithms usually perform in linear time with respect to the size of the input (i.e., they are O(n) where n is the size of the input),[57] but their performance is sub-optimal. There are even some incremental, linear-time, and optimal algorithms, but which are inappropriate in cognitive science because they are based on *ad hoc* heuristics and the setting of a large number of parameters. An extreme example of the use of heuristics is an algorithm known as "ISODATA" (Ball and Hall, 1965), that manages to perform optimally and in linear time, but is so complicated that it is safe to say that it exists only as a module in certain FORTRAN libraries of clustering and data mining routines. An additional problem for algorithms packed with heuristics is that often their advantages can be exploited only when the size of the input is large (e.g., on the order of thousands). The reason for this is that heuristics often require an initial computational overhead that imposes a significant burden if the size of the input is small. Cognitive systems, by contrast, often have to form categories on the basis of a very small number of examples, as in Figure 8.5.

8.3.2 Phaeaco's group-formation algorithm

Phaeaco's algorithm accepts as input a set of "exemplars" (for example, they could be the set of dots shown in Figure 8.5, or entire representational structures such as those presented in chapter 7), declared as a parameter of type "queue" in Figure 8.6, to emphasize the idea that a single scan of the exemplars is enforced. The algorithm constructs and outputs a set of "patterns", each of which is a statistical summary of a group formed by the exemplars. The algorithm is also called "Patterns", for lack of a more suitable one-word mnemonic.

---

[57] In contrast, non-incremental algorithms are usually O($p(n)$), where $p(n)$ is a polynomial of degree $n \geq 2$, because they scan the input repeatedly.

```
Algorithm Patterns (queue exemplars)
    set patterns ← empty
    set known-exemplars ← empty
    for each ex_i in exemplars do the following
        if ex_i is not Similar to one of patterns then
            real max-similarity = 0
            pattern closest ← null
            for each ex_j in known-exemplars do the following
                real similarity = Match (ex_j, ex_i)
                if similarity > max-similarity then
                    max-similarity ← similarity
                    closest ← ex_j
            if max-similarity > clustering-threshold
                pattern new-pattern ← Form Pattern out of closest and ex_j
                if new-pattern Resembles one of patterns then
                    discard new-pattern
                else
                    Add new-pattern to patterns
                    Remove closest from known-exemplars
        Add ex_i to known-exemplars
    output the Union of patterns and known-exemplars
```

Figure 8.6: Phaeaco's basic group-formation algorithm

The notation used as pseudo-code in the algorithm of Figure 8.6 should be self-explanatory: indentation signifies blocks of code in the usual programming convention, keywords of the hypothetical language are shown in bold type, identifiers (variables) in italics, types of identifiers in bold italics, and calls to sub-algorithms (to be discussed below) in regular type, but capitalized.

The algorithm starts by first initializing its output set of patterns and a local set of "known exemplars" to the empty set. Each input exemplar is compared against each pattern in the current set of patterns. If it is not "similar enough" (this will be explained next) to any of the current patterns, then the following happens: before adding the input exemplar to the set of known exemplars, each of the

known exemplars is considered as a candidate to form a new pattern with the input exemplar. If this new pattern is *really* new, i.e., if it does not match well with any of the current patterns, then it is added to the set of patterns; otherwise it is simply discarded. Finally (when all input exemplars have been considered), whatever known exemplars have been left out (i.e., those that did not form a pattern with any of the input exemplars) are turned into "groups made of a single element", and are added to the set of patterns.

The meaning of the last step can be explained by the example in Figure 8.7: when the algorithm ends, the two isolated dots are left in "known exemplars", so each can be termed trivially a "group" and added to the set of patterns.

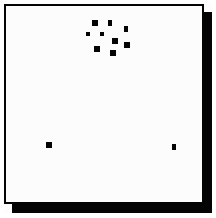


Figure 8.7: Another box from BP #166, with one group (pattern) and two isolated exemplars

Features of the Patterns algorithm include the use of a "clustering threshold" and the storing of input exemplars in the temporary list of "known exemplars". The clustering threshold is a parameter that can be considered as a near constant, but can vary slightly depending on contextual pressures: increasing its value decreases the "tolerance" of how different two exemplars can be before they are put in different groups. Storing input exemplars in the set of known exemplars makes this algorithm quadratic ($O(n^2)$) — more in §8.3.4.

The algorithm of Figure 8.6 is only the "main" procedure of an entire set of sub-algorithms that are explained in what follows, starting with one implied by the line that reads: "**if** ex_i is **not** Similar to one of *patterns*…".

```
Algorithm ExemplarIsSimilarToAPattern (pattern exemplar, set patterns)
    if patterns is not empty then
        real max-similarity = 0
        pattern closest-pattern ← null
        for each pattern in patterns do the following
            if pattern Resembles the exemplar then
                real similarity = Match (pattern, exemplar)
                if similarity > max-similarity then
                    max-similarity ← similarity
                    closest-pattern ← pattern
        if closest-pattern is not null then
            Update closest-pattern with exemplar
            return true
    return false
```

Figure 8.8: Algorithm testing for similarity of exemplar to patterns

The remaining sub-algorithms are straightforward:

- To "Match" two exemplars means to compute their similarity as specified in §8.2, and to return the value as a real number.
- The decision "Resembles" (which appears in both listed algorithms) is implemented by simply calling "Match" and testing whether the resulting value of similarity is greater than the clustering threshold (the same threshold that was used in the main procedure of Patterns).
- Finally, the lines "Form Pattern" (out of two exemplars) in Figure 8.6, and "Update" (a pattern with an exemplar) in Figure 8.8, are essentially calls to the same sub-algorithm, which causes the exemplar to update the statistics of a pattern, and which merits an explanation in a subsection of its own.

8.3.3 Pattern updating

The preceding discussion has explained how to compare either two exemplars, or a pattern (the statistical summary of exemplars) with an exemplar, or two patterns, thereby obtaining a measure of their difference, and how to use that difference to decide that exemplars belong to a group, represented by its corresponding pattern. But it has not explained how a pattern is constructed in the first place.

The construction of a pattern out of two exemplars follows a "recursive" procedure carried out on the representational structure, with all the same disclaimers against deep and blind recursion as were expressed in §8.2.4. At the base of this recursion stands the statistical updating of featural nodes.

Suppose two nodes that represent slopes of line segments that belong to different exemplars are given. For example, the line segments might be the left slanted sides of two "A"'s, printed in two different fonts (Figure 8.9). The first slope might be 65%, and the second slope 75%. If we try to generalize and think of the two instances (exemplars) of the letter "A" as "one idea", then we can say that the average slope of the left side, given only these two exemplars, is (65 + 75) / 2 = 70%. A standard deviation can also be computed, and it is $65^2 + 75^2 - 2 \cdot 70^2$ = 50% (see Equation 8.10).

Figure 8.9: The letter "A" in two different fonts

In general, if we already have a statistical sample with mean value $\bar{x}$ and size *N,* then the formula for computing the new mean $\bar{x}'$ after inserting an additional datum *x* in the sample is:

$$\bar{x}' = \frac{N \cdot \bar{x} + x}{N+1}$$

Equation 8.9: New mean given an additional datum, old mean, and sample size

If the old sample has standard deviation *s,* then the formula for computing the new standard deviation s' after inserting x in the sample uses the new mean $\bar{x}'$ from Equation 8.9. It also uses the sum of squares of the sample data, which is denoted by Σ_2 in Equation 8.10:

$$s' = \frac{\Sigma_2 + x^2 - (N+1)\bar{x}'^2}{N}$$

Equation 8.10: New standard deviation given an additional datum and new mean

The quantity $\Sigma_2 + x^2$ is the sum of squares of the new sample. This quantity makes evident the reason for keeping the sum of squares explicitly as a field in the statistical structure of a feature in Table 7.1: to avoid storing explicitly the sample data x_i. Thus, for the sake of completeness, four more formulas must be added to show how the rest of the fields in Table 7.1 are updated after adding a new datum, including the new minimum and maximum value of the sample:

$$N' = N+1$$
$$\Sigma_2' = \Sigma_2 + x^2$$
$$Min' = \min(Min, x)$$
$$Max' = \max(Max, x)$$

Equation 8.11: New number of data, sum of squares, minimum, and maximum value

All the equations above are easily generalized for the case where the added quantity is a statistical sample with N > 1, instead of a single datum.

Thus, suppose several exemplars have been encountered, as in Figure 8.10:

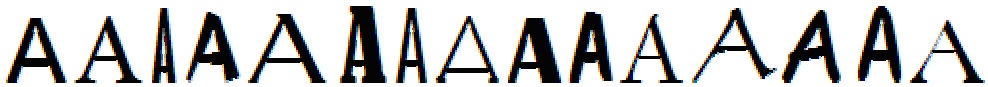


Figure 8.10: A larger sample of input exemplars of "A" 's in various fonts (Figure 2.2, repeated)

Then, assuming Phaeaco has succeeded in placing them all in one group, the formed pattern that represents the statistical summary of the exemplars in the group will contain nodes for features that will correspond to the average value, standard deviation, etc., of the left slanting line, its average length, and so on for each feature of the pattern. To depict concretely something that could be the result of this pattern, consider the drawing in Figure 8.11.

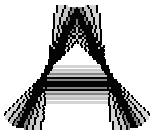


Figure 8.11: Concrete (and oversimplified) depiction of what a pattern can generate

Observe, however, that Figure 8.11 is only a rough approximation of what could be generated from the true structure of the pattern stored in Phaeaco's memory. The pattern itself is nothing like a "template bitmap" (as might be erroneously inferred from Figure 8.11), but has the structure of a representation, similar to the structures of chapter 7, except that its featural nodes are full statistical samples with the structure of Table 7.1. (A pattern for something like the letter "A" is too complicated a structure to be depicted in a single page of this thesis.)

If the statistics of the pattern are updated by the equations given earlier, then how is the representational structure of the pattern constructed? Even in a case as seemingly simple as two exemplars of the letter "A", a question arises about how

small structural discrepancies are to be summarized in the pattern. For example, serifs might be present in some of the "A"'s but absent in others, as in Figure 8.10. Should the pattern include serifs in its structure, or not?[58]

The answer in Phaeaco's architecture is literally "yes and no". This equivocal result is achieved by modifying an element of representational structures that has not been mentioned yet: the strength of links between nodes. Links in patterns have strengths that exactly follow the rules for how activations increase and decrease their values, as described in §7.2.

The idea is simple: when a new structural element (such as a serif) is encountered, the element is linked into the structure of the pattern, but the strength of its link is weak. Each new exemplar that contains this element increases the strength of the corresponding link. As long as patterns are formed in the Workspace, links between nodes do not decrease; but when a pattern is copied to the LTM, as will be explained in the next chapter, all links can weaken (or be enhanced) over time, by mechanisms that are specific to the LTM.

This observation also answers the question of what the pattern built from seemingly incongruous exemplars, such as a triangle and a square, would be. If contextual pressures dictate lumping together triangles and squares (e.g., with the concept "polygon" highly primed in LTM), then the structure of the pattern will include four λ-nodes, but these nodes will be linked in peculiar ways, representing the ways line segments are connected in both a triangle and a square; additionally, some link strengths will be weaker than others.

---

[58] No attempt is made here to propose a pattern structure that would represent artistic designs of the letter "A" that do not consist of three lines, possibly with some serifs (i.e., typically in Times Roman font). For a treatment of this topic see the Letter Spirit project (Rehling, 2001).

8.3.4 Comparison of algorithms

To ascertain that the Patterns algorithm is not deficient in comparison with other known clustering algorithms, its performance was measured against three algorithms that are at least as easy to implement:

- The "*k*-means" algorithm (McQueen, 1967):
    1. Initialize the first *k* clusters with random exemplars from the input.
    2. Distribute the exemplars among the present clusters; in the process, update the cluster centers.
- The "leader" algorithm (Hartigan, 1975):
    1. Assign the first exemplar to the first cluster.
    2. Assign the next exemplar to the most similar cluster, or create a new cluster if the similarity exceeds a given threshold.
    3. Repeat step 2 until all exemplars have been assigned to a cluster.
- The "Lu and Fu" algorithm (Lu and Fu, 1978):
    1. Given the current exemplar, assign it to the cluster that includes the exemplar that is closest to the given one.
    2. If the similarity of the closest exemplar exceeds a given threshold, then create a new category for the current exemplar.
    3. Repeat step 1 until all exemplars have been assigned to a cluster.

The first two algorithms do not store the input exemplars for re-examination, whereas the third one, like Patterns, stores exemplars and re-examines them. The algorithms were tested on a variety of clustering cases, varying the number of clusters, the number and density of exemplars in each cluster, and the distance among clusters (how well separated they were). Exemplars were real numbers in [0, 1]. A "performance index" was established to automatically assign a grade of

success to each run of each algorithm, with values in the range [0, 100], with 100 meaning "perfect success". It was possible to compute this index automatically because the tested exemplars were always forming groups unambiguously. For those algorithms requiring a clustering threshold (i.e., all except *k*-means), the value of the threshold at which the algorithm performs best was selected. The results are shown in Table 8.1.

Algorithm	**Performance index** (0–100)	**Standard deviation**	**Clustering threshold**	**Threshold tolerance**
Patterns	98.5	7	0.83	0.78–0.88
leader	92.9	16	0.82	0.78–0.85
Lu & Fu	78.8	28	0.85	0.81–0.88
k-means	54.7	36	—	—

Table 8.1: Comparison of algorithmic performance

The performance index was averaged over a large number of runs, so the third column gives the standard deviation of the sample of measured indices. All three algorithms that include a clustering threshold showed some tolerance to the value of their threshold, which means that the specific (best) value listed in column "Clustering threshold" is not crucial for the algorithm to perform well.

Further tests in speed and memory usage showed empirically what was expected theoretically: the leader and *k*-means algorithms are the fastest and use the least memory among the four, since they perform a single pass over the input exemplars. Apparently, they achieve this at the expense of performance. (Nonetheless, the performance of leader is only very slightly inferior to that of Patterns.) The speed and memory results are summarized in Table 8.2.

Algorithm	**Speed**	**Memory**
k-means	0.38	O(*n*)
leader	0.49	O(*n*)
Patterns	0.92	O(n^2)
Lu & Fu	1.38	O(n^2)

Table 8.2: Comparison of algorithmic speed and memory requirements

The values shown in the "Speed" column of Table 8.2 are in milliseconds averaged over 6000 repetitions, but these numbers have only relative, rather than absolute, significance.

Overall, the Patterns algorithm compares favorably in performance with the other three algorithms, although it achieves this at a sacrifice of speed and memory in comparison to the leader algorithm. However, the observed sacrifice occurs only when a large number of exemplars must be clustered quickly, which usually is a requirement in computer science applications. Cognitive categorization tasks are usually slow when they are sequential (nobody asks people to cluster thousands of items in seconds), or fast when parallel, as in the example of dot-grouping in Figure 8.5. In the latter case, Phaeaco, with its low-order polynomial-time algorithm (O(n^2)), implemented on a sequential processor, is "competing" against our human neurally-implemented parallel mechanism. This is not a handicap from the point of view of the theory of computational complexity, according to which a parallel algorithm of constant time can at best be of polynomial (linear) time if implemented sequentially (e.g., Papadimitriou, 1994, p. 139).

8.4 Pattern matching as the core of analogy making

Every scientific discipline has its "core" or "founding principles", and until such principles are spelled out explicitly and become widely accepted by the practitioners in the field, the field is considered insufficiently scientific, a mixture of art and science. For example, among the founding principles of physics are Galileo's and Newton's laws of motion; in chemistry, Dalton's atomic theory and Mendeleev's construction of the periodic table of elements; in biology, Hooke's discovery of cells (and coinage of the word), Darwin's theory of evolution by natural selection, and Watson and Crick's discovery of the structure of the DNA molecule; in cosmology, Hubble's discovery of the expansion of the universe and the subsequent Big Bang theory; in mathematics — if we interpret the notion "science" in a broader sense — the axiomatic method, systematized by Euclid; and so on. But what are the founding principles of cognitive science?

Many cognitive scientists have attempted to discover the "core" of their field, but not all such attempts can be regarded as successful. In the 1970's and early 1980's, when the term "cognitive science" was not yet in common use (and when most of the related work was done under either cognitive psychology or artificial intelligence — two mutually non-communicating fields), some researchers proposed that "problem solving", and searches through exponentially large spaces with the use of heuristics, are fundamental. It soon became evident, though, that these are principles for building computer systems that can solve only in an alternative, computeristic way problems traditionally tackled by human minds. This idea may be said to have culminated with the construction of the chess-playing program Deep Blue and its win over the human world chess champion (see more in §4.4.2). Such methods, however, have little or no relation to cognitive principles. At the same time there was strong support for the idea that

mathematical logic is at the core of the cognitive engine of every intelligent agent, and the repercussions of this view can still be felt in various publications and research programs. Nonetheless, the view that this assumption is erroneous, and that the predicate calculus is an epiphenomenon of human cognition rather than a foundational "pillar" has already been expressed in this text (§7.4.10).

The 1990's saw a resurgence of research in artificial neural networks, and — helped by discoveries of techniques in exploring the blood flow in the human brain — a wealth of observations of areas of the brain while it engages in various cognitive tasks. Although no one can deny that human cognition is ultimately based on neural mechanisms, it is questionable whether neurons constitute the sole and ultimate "currency unit" of cognition. The primary subject of cognition is not the brain, but the mind. (Not even specifically the human mind, but the Mind, abstractly.) Brains are the biological organs that evolution selected in animals to cope with properties of their environment. But brains (and neurons) do not necessarily imply minds. For example, chickens have brains, too, but their cognitive abilities are not exactly at the cutting edge of research in cognitive science. Not even the quantity of neurons qualifies as what makes the difference between full human intelligence and its approximations: whales, most dolphins, and elephants, all have more neurons in their brains than do humans, but their cognitive abilities are at the level described in psychology as "animal cognition". Also, mentally retarded humans, though significantly more intelligent than any animal species, are incapable of serving as examples of an average human mind.

No neurobiologist is yet in a position to tell which brain belonged to a mentally retarded person and which to a genius after a postmortem examination.[59]

One of the fundamental tacit assumptions in cognitive science has been that cognition can be described abstractly as a set of principles, and is implementable in ways other than the only example of which we are aware, i.e., in biological, neural hardware. Thus, even though neurons are sufficient for minds, they are not necessarily necessary. If cognitive scientists did not hold this belief, much of the motivation for building intelligent machines would not be justified.

Thus, neurons are the building blocks of a complex system (brain) that gives rise to the subject of cognitive science but is not *the* subject of it; it is the subject of a related discipline (neurobiology). Research on the computeristic counterparts of neurons (artificial neural networks, or ANN's) represents efforts to show that some specific cognitive problems are solvable within the particular framework of ANN's, implying that they are also solvable in the framework of real neurons (which we already know), and that possibly they are also solved in the same way by neurons (which is at best a conjecture). But research in ANN's does not necessarily address the core principles of cognition. ANN's might, for all we know, be an attempt to work in cognition at the wrong level, akin to trying to understand biological principles in terms of simulated chemical reactions.[60]

There has been a different idea for what stands at the foundation of cognition. According to this idea, championed most notably by Hofstadter since the early 1980's, the core of cognition is "analogy-making". Most people — cognitive scientists and laypeople alike — upon hearing this term, recall salient examples of

---

[59] Certain conditions, such as Down's Syndrome, can be identified by examining characteristics of the DNA structure (the number of chromosomes), but this apparently is unrelated to the structure of the brain itself — any cell with a nucleus from the tested individual would suffice.

[60] Every good biologist needs to have a good knowledge of chemistry, but knowing chemistry alone does not make a good biologist.

analogies. For instance, scientists (but not physicists or chemists) might think "an atom is like a planetary system"[61] is an example of an analogy. Laypeople might choose examples such as: "Afghanistan was the Vietnam of the Soviet Union", or ones with predictive power, such as: "Iraq will turn out to be the new Vietnam for the U.S." Hofstadter himself has enriched the repository of creative analogies by bringing up amusing examples, such as: "Cambodia is the Vietnam of Vietnam", or "Dennis Thatcher is the First Lady of Britain".[62] Right… but what does all this have to do with the foundations of cognition?

It turns out that when Hofstadter uses the term "analogy-making" he means it in a much more general and fundamental sense than the one suggested by such humorous examples. In particular, he would say that analogy-making is to see two tables (e.g., a formal, stylish, dining-room table, and a casual, kitchen-room table), and perceive their abstract common structure; or, to see two apples and perceive them as instances of the same kind of fruit; or, to see a pen and perceive it as "a pen", i.e., categorize it under the already known concept of "pen". The last example implies that any instance of perception and subconscious categorization is an instance of analogy-making, in Hofstadter's use of the term. Indeed, in "Analogy as the Core of Cognition" (Hofstadter, 2001), we read:

> The process of inexact matching between prior categories and new things being perceived (whether those "things" are physical objects or bite-size events or grand sagas) is analogy-making *par excellence.*

---

[61] A wrong analogy, but naturally it persists in the minds of scientifically educated people, given the number of old physics textbooks depicting an atom as a planetary system. But given the wrong model, it is a great analogy.

[62] This was said at a time when the prime minister of Britain was Margaret Thatcher.

In Phaeaco's terminology, this sort of analogy-making is called pattern-matching and categorization. There is no need to define an artificial border between pattern-matching and analogy-making (as the latter is understood by most people), because there is no use that such a border can have: it is the same mechanism, all the way from "grand sagas" to physical objects. But in spite of its being the same mechanism, the use of different terms for its two "extremes" is justified, because it facilitates communication. An analogy will help explain the justification for using both terms, "pattern-matching" and "analogy-making".

When an atom of carbon (C) joins with four atoms of hydrogen (H) to make a stable molecule of methane (CH_4), we say that the carbon "combined" with hydrogen. But when people consume a sandwich, we say they are "eating" it. People are not said to "combine" with their sandwich, nor is carbon said to "eat up" hydrogen. Yet, deep down it is the same mechanism: the process by which we evolved from organic molecules to complex mammals involves an astronomically large number of chemical exchanges.[63] But even though the mechanism remained the same in principle, it became so complex that we feel we must reserve the verb "to eat" for when we consume our food, rather than use it to describe even simple combinations of atoms and molecules.

The previous analogy is deeper than might be perceived at first thought. Just as food consumption in the world of biology has its chemical analogue in molecule combination, so the cognitive mechanism of analogy-making has its biological analogue in pattern-matching. Animal cognition varies greatly in

[63] Somewhere along this process (early on, presumably in the first half billion years of Earth's history) organic molecules "learned" to do something that approximated replication, and then gradually perfected this achievement, entering the biological stage in our planet's history. But no matter which stage we look at, chemical or biological, the mechanism of chemical combination or food acquisition — which description sounds more suitable depends on the stage — has remained identical throughout our evolutionary past.

degree, from the absolute zero of sponges, to the highest degree of which we are aware: the cognition of our own species. Although it is still largely a mystery whether animals other than humans maintain rudimentary mental representations, it is plausible that pattern-matching was not discovered suddenly by human beings, but existed at least among our lineage of ape ancestors, and probably among other cognitively complex animals as well. Are we to believe that when we see a pen we perceive it as "a pen", or "a writing instrument", but when lions see (or smell) a zebra they are unable to perceive it as "a zebra", or "food"? It all depends on how complex animal representations can be. Conceivably, Skinnerian stimulus-and-response behaviorism has its place among creatures that appeared early on in evolutionary history and changed little (if at all) since, but is insufficient to describe the cognition of animals that developed complex cognitive abilities and behaviors. It is an example of anthropocentric chauvinism to suppose otherwise.

In summary, our ability for analogy-making (at its best), or pattern-matching (at its humblest) — whatever name we give to it — is the fundamental ability of cognitive creatures to perceive the world and make sense out of it, by assigning each object to a known category; to form categories by being exposed to sufficiently similar objects; and even to perceive the objects themselves, which is a prerequisite for categorization. How do we manage to see "objects" in the world, rather than random collections of "pixels" sent to our visual cortex through the rods and cones of our retinas? We do it because some collections of "pixels", due to spatial proximity (such as the dots in Figure 8.5), or proximity due to other features (color, texture, etc.) seem to "belong together". By making groups out of what seems to belong together, we perceive "objects".

Note that the use of "we" in the previous paragraph does not imply that objects are only an artifact of cognition. Objects must exist objectively in the world; animals simply evolved to perceive them, as the previous paragraphs on the evolution of pattern-matching and analogy-making suggest. The present work can be regarded as an existence proof of the proposition that minds are not necessary to perceive and thus verify the existence of objects. After all, Phaeaco can perceive them, too.

CHAPTER NINE

Long-Term Memory and Learning

9.1 Motivation: is LTM really necessary?

Previous FARG systems (§6.2) lack a true LTM (of the kind that is saved on a permanent medium, such as a computer disk, and becomes available again at the next instantiation of a program). Systems such as Copycat, Tabletop, and Letter Spirit include in their architecture the Slipnet, whose nodes become activated whenever information is processed and whose links shrink or expand according to their "elastic" nature. But when the system stops running, all activity and whatever modifications occurred in the Slipnet are swept from the computer's memory: the next time the system is invoked, it behaves as if the previous session had never occurred, since the Slipnet nodes start anew with their "factory defaults", to use the terminology of the software industry. Metacat includes a memory which is "longer term" compared to other FARG systems (though still not saved permanently), but its memory is organized as a database of files with facts regarding previous sessions, rather than as a true cognitive memory in Phaeaco's sense (to be explained in this chapter).

At least theoretically, a memory-resident Slipnet is all that is required by a cognitive system: every modification effected permanently on a system that periodically saves its Slipnet (or whatever its LTM is called) on disk can also be effected in an identical way if the same system undergoes the same learning

experiences in a single session. There is no theoretical advantage to be gained by temporarily suspending the operation of the system, "waking it up" some time later, and restoring it more-or-less to its previous state, the one that existed before "going to sleep". The periodic "sleep and wake up" stages that a true LTM allows simply prolong the training period of the system, according to this view.

Although there can be no valid theoretical argument against this idea, some practical considerations are worthy of attention. The expectation that a system would undergo its entire training period in a single session, and thus learn a significant amount of knowledge in a highly concentrated timeframe, is tantamount to the idea that a person could go through their childhood and years of formal education in a "fast forward" manner, compressing into a matter of minutes (the usual duration of a computer session) a significant part of many years of experience. It is common knowledge that the time it takes people to learn various subjects generally cannot be compressed significantly, but instead time must "take its course" until knowledge "settles". This can be attributed to the slowness of the underlying hardware medium (neurons). It is conceivable that the advent of faster computer hardware will trivialize this practical problem, compressing years of experience into minutes. But some simple calculations suggest that this will not happen any time soon. To compress only 15 years of experience into 15 minutes (which is at the limits of what can be considered a reasonable time to wait before a program trains itself and becomes available) one needs a computer that runs approximately ½ million times faster than present ones.[64] Assuming an improvement in computer speed on the order of 1.5 times

[64] There are 525,600 15-minute intervals in 15 years. The implicit assumption is that a computer would be required to undergo experiences similar to those of a person in the first 15 years of life, which is a moot subject.

per year,[65] we conclude that we must wait for more than 32 years[66] before technology grants us this wish. There must be a more practical way to train a computer.

There is an additional problem. Phaeaco's LTM has a feature lacking from all Slipnet-like memories of FARG projects. In addition to the "elasticity" of links that causes them to act like rubbery bands changing the distance between nodes, Phaeaco's links can have their elasticity permanently modified, all the more so if they shrink and extend repeatedly. As will be explained in §9.3, it is as if the frequent use of memory gradually causes links not only to shrink more than before, but also to "harden", making it easier for activation to spread through the linked nodes, and harder for the links to go back to their "factory default" length. These are all features of Phaeaco's "learning" abilities (§9.5), and for them to have any perceivable effect on the behavior of the system, a significantly long time must pass, during which the system gains experience. This property of Phaeaco's LTM makes it more difficult to insist that memory be updated with "a life's worth of experiences" in a single training session.

Finally, as will be explained in §9.4, the periodic sleep-and-wake-up stages facilitate the implementation of a form of forgetting. Forgetting can be seen as "culling" — allowing only the salient memories to remain accessible — which is, after all, another way of learning.

---

[65] This rate of improvement in hardware speed, known in the computer industry as "Moore's law" (in reality a self-imposed industry marketing strategy, rather than a true law of nature), was already diminishing by the time the present text was written (2006).

[66] This is the number of times that 1.5 must be multiplied by itself to yield approximately ½ million.

9.2 From visual patterns to concepts

Phaeaco's memory is populated with visual patterns (chapter 8), which are called "core structures of concepts" in the context of the LTM. The reason for this choice of terms will become evident in §9.2.1. The change in terminology (from "patterns" to "concepts") is necessary because, as will be further discussed in §12.1, visual memory forms the basis for the more abstract conceptual memory that seems to be the hallmark of human cognition. Thus, "galaxy" and "running" can be thought to invoke visual patterns, but "democracy" and "theory" hardly do so, and are better described as "concepts". Although Phaeaco's present implementation supports no more abstract concepts than visual patterns (that were once formed in the Workspace), the hope is that future work will use the present framework to move naturally into the domain of abstract concepts. Accordingly, the network of entities that forms the LTM in Phaeaco is not called a "Slipnet", but a "conceptual network".

If the LTM consists of concepts, when do patterns migrate from the Workspace to become LTM residents and be christened "concepts" (or, more precisely, "core structures of concepts")? The answer is: as soon as the visual input that was responsible for their creation ceases to exist. In the case of solving a BP, this happens as soon as the BP is solved, whereas in the case of receiving input from a Mentor session (§5.2), it happens as soon as the input in the Mentor's box has been "seen" and a representation of it has been created.

Patterns created in the Workspace from visual input are not simply stored in the LTM, adding one more resident each time, but matched against existing concepts, using the same mechanisms that were described in §§8.2–8.3. If the new pattern is found to match some LTM concept sufficiently closely, the concept is

simply updated with the pattern (according to the equations of §8.3.3). But if the pattern is different from all known LTM concepts, it will form the foundation of a new concept. This idea is essentially the same as the idea that groups of exemplars form visual patterns in the Workspace (§8.3).

If the LTM consists of an "ocean" of concepts, there must be some practical way to navigate this ocean to find which concepts fairly closely match a given pattern (which is about to migrate to the LTM) without testing each concept in turn. Indeed, an indexing scheme that facilitates access to concepts in LTM will be outlined in §9.4.1.

9.2.1 A slight departure from the Slipnet concept of concept

Figure 6.3, taken from GEB, is a depiction of a portion of a Slipnet. Each node, as explained in §6.2.1, serves as the core of a concept, and nodes that are in close proximity are said to belong to the halo of that concept.

Phaeaco's conceptual network is slightly more complex than a Slipnet. A concept in Phaeaco usually has a structure, unless it is a primitive. For example, "triangle" is a non-primitive concept, and chapters 7 and 8 explained how the pattern of a triangle can be formed, and hence can migrate into LTM, either forming the basis for a new concept if nothing similar exists, or updating a similar concept. But a triangle has a structure: it is made of three line segments, among other parts. In Phaeaco's LTM, each of the line segments has a node that corresponds to it (which is not any specific line segment drawn in a box, but "a line segment", one of three that constitute a triangle). Each of these three line segments is a node separate from the node that represents the Platonic concept "line segment", which is a more abstract idea than the notion "line segment (or side) of a triangle". In Figure 6.3 there is a node for "line segment", and a link that connects "triangle" and "line segment" with a relation labeled "composed

of". But how is the individuality of each of the three sides represented in a Slipnet? How can a single "line segment" node represent the particular way in which the sides of a triangle meet at three vertices, which is different from the way the three line segments of the letter "A" meet and touch each other?

As the foregoing shows, the concept "triangle" in Phaeaco is not just a node labeled "triangle" (the core) surrounded by a loose association of proximal nodes (e.g., "rectangle") that are part of its halo, but is also a structured collection of nodes. The notion "halo" is still valid in Phaeaco and has the same meaning as in a Slipnet, but the notion of "core" is modified to include not just a single node, but the entire collection of nodes that make up the structure of the concept. To avoid any possible conflict with the term "core" as used in earlier FARG systems, the term "core structure" will be used to refer to the structure of a concept in Phaeaco.

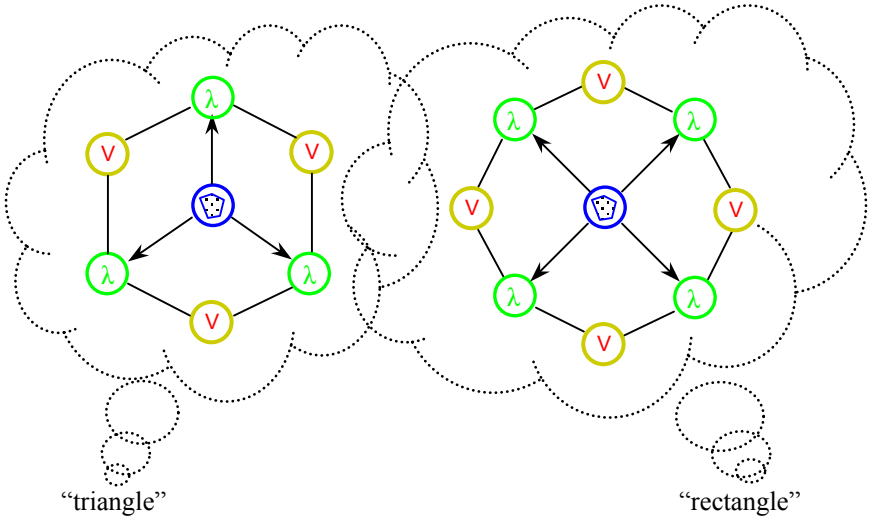


Figure 9.1: Core structures of concepts "triangle" and "quadrilateral" in LTM

Figure 9.1 shows two concepts, "triangle" and "rectangle". The cloudy regions that surround the core structures represent the halos of these two concepts

(note that they overlap). The core structures are by necessity shown with a bare minimum of nodes among those that would be included in reality.

A question immediately arises: is the Platonic node "line segment" (which stands outside of the "triangle" core structure) *not* in the halo of "triangle"? Recall that in the Slipnet of Figure 6.3 there is a direct connection between the nodes "triangle" and "line segment" (the Platonic notion), which is missing from Figure 9.1. The answer is that now "triangle" is linked indirectly to the Platonic "line segment" node, because each of the λ-nodes of the core structure of a triangle is of type "line segment". Figure 9.2 adds the Platonic "line segment" into the picture, somewhere in the common halo of "triangle" and "rectangle".

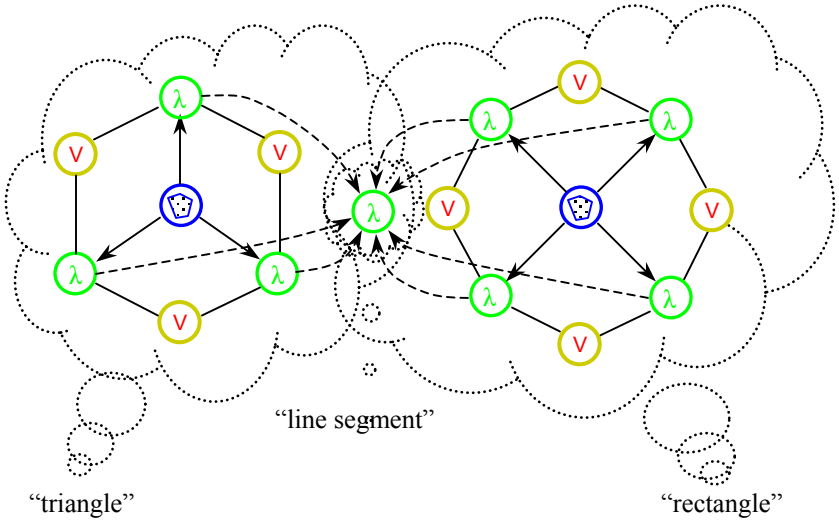


Figure 9.2: Platonic "line segment" added in Figure 9.1

A link from each λ-node of the two core structures to the Platonic "line segment" is shown in Figure 9.2. These links represent the relation "is of type". Similar links connect not only λ-nodes but every node of a core structure with its corresponding Platonic primitive concept.

9.3 Properties of LTM nodes and connections

9.3.1 Long-term learning of associations

How do we learn to associate notions or procedures with each other, initially with difficult conscious efforts, but later with the feeling that the task is becoming easier, until eventually it is automatic and vanishes from consciousness? For instance, anyone who has tried to learn to speak a foreign language has had the experience of words "coming to the tongue" progressively more easily. A non-native speaker who learns to self-correct a linguistic error, after being corrected by a native speaker, might experience the same feeling of gradual automation. Mechanical tasks, such as learning to drive and tying shoelaces or a tie, also pass through similar stages. In the visual domain we often need to acquire a mental map before we can navigate confidently in a previously unfamiliar neighborhood or town. We acquire the map progressively, after repeatedly traveling through the region. In BP's, a solver might learn to imagine the convex hull of irregularly shaped objects, after being repeatedly exposed to BP's that employ this notion in their solution. In this case, the learned association is: "irregular shape → think of convex hull".

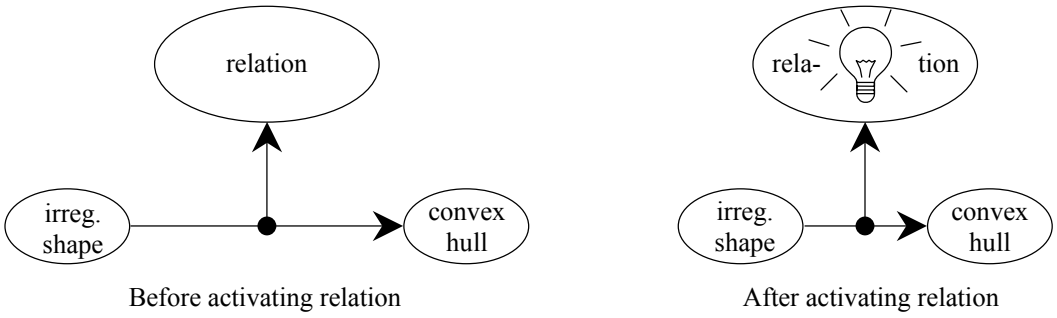


Figure 9.3: Easing the spreading of activation in Slipnet

How is this observation modeled in a "traditional" Slipnet? When the relation between two Slipnet nodes is activated, their link can be thought of as shrinking. This has been illustrated in Figure 6.7, which is repeated in Figure 9.3, but adapted to the example: "irregular shape → convex hull". The smaller conceptual distance allows more activation to spread from "irregular shape" to "convex hull", making it easier for the latter idea to follow from the former one.

But this cannot be an accurate model of long-term learning, because the increased activation of the node labeled "relation" in Figure 9.3 is understood to be a relatively temporary event. The elastic nature of the Slipnet links implies that activations not only increase but also decrease; otherwise, concepts would be permanently activated in memory — a rather un-cognitive-like situation. Indeed, in FCCA we read (Hofstadter, 1995a, p. 212, emphasis in the original):

> "The Slipnet is not static; it dynamically responds to the situation at hand as follows: Nodes *acquire* varying levels of activation (which can be thought of as a measure of relevance to the situation at hand), *spread* varying amounts of activation to neighbors, and over time *lose* activation by decay."

Thus, the conceptual distances between related nodes not only shrink, but also relax while activation decays. But if they do relax, do they return to their original "factory default" lengths? In previous FARG projects, the answer has been "Yes", implying that Slipnet activations do not model truly long-term knowledge, but rather what the word "activation" implies, i.e., "respond to the situation at hand", as mentioned in FCCA. The architecture of Phaeaco's LTM modifies the original Slipnet model to accommodate long-term learning, as described below.

The basic idea is that when activation decays, links should not return to their original lengths; some residue of the earlier shrinking should remain on the previously activated node (i.e., the node labeled "relation" in Figure 9.3), acting as a reminder that the links with lengths that depend on this node were shrunk a short while ago. Residues should have a cumulative effect, so that if, for example, the activation of a node repeatedly reaches its highest value, the overall residue should "remember" this fact. Nonetheless, the accumulated residue cannot be a static quantity that remains fixed once formed; after all, minds hardly retain memories forever (more on this issue later). Thus, the residue must also decay, but at a *much* slower rate than the activation itself, reflecting the slowness in the loss of long-term memory.

The idea of an activation residue is implemented in Phaeaco by a quantity called "significance", which is added to the current activation, resulting in an overall "weight" of the node. Thus,

$$\text{weight} = \text{significance} + \text{activation}$$

The significance of a node is the long-term component of its weight, and possesses the same architectural properties as an activation, introduced in §7.2. The activation is the short-term component of the weight, functionally identical to activations in a Slipnet (e.g., it spreads to neighboring nodes), and structurally described also in §7.2. The only difference between significance and activation is in their rates of increase and decay. Specifically, the increase and decay rates of significance are several orders of magnitude slower than the corresponding rates of activation. Their sum, i.e., the weight, is what in a Slipnet "can be thought of as a measure of relevance to the situation at hand" (see the passage quoted earlier). The weight is also the quantity w_i in Equation 8.7 that determines the computation

of the psychological distance between two representations (§8.2.3). The way in which the increase in activation affects the significance is explained next.

Recall that activations increase and decrease according to a sigmoid function *f,* discussed in §7.2, and depicted again in Figure 9.4.

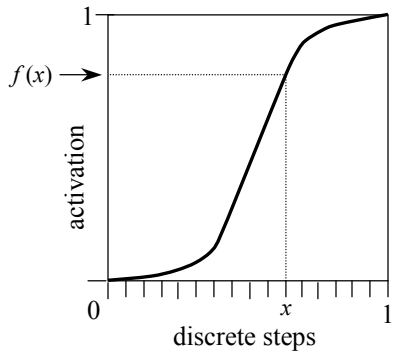


Figure 9.4: The sigmoid function *f* of an activation (Figure 7.17, repeated)

Recall also that the number of discrete steps on the *x*-axis that determine the domain of *f* is a parameter of the activation. Call this parameter the "stability" of the activation. The justification for choosing this term is the following: assuming it takes a fixed interval of time for the activation to decay from step x_2 to its immediately previous step x_1 — hence, for the activation value to drop from $f(x_2)$ to $f(x_1)$ — if the parameter "number of discrete steps along the *x*-axis" is large, then the activation value takes longer to decay than if the same parameter is small; hence the notion of "stability" of an activation. As was implied by the discussion in the previous paragraphs, the stability of the significance of a node or link is several orders of magnitude greater than the stability of the activation of the same node, or link.

An additional parameter is the "limiting activation". This is a number $l > 0$, very close to zero, the meaning of which is that the activation must remain between l and $1 - l$. Thus, l defines simultaneously a lower and an upper bound for the activation value. Like stability, the limiting activation is not a system-wide but a per-activation parameter, i.e., each activation has its local value of limiting activation.

What connects these parameters with the notions of activation and significance is that each time the activation value of a node or link reaches its maximum ($1 - l$), the significance of that node or link receives a boost, increasing to the next discrete point along the x-axis. But also, when the significance itself reaches its maximum, its stability is increased by a small amount. Thus, repeated maximizations of the short-term component (activation) not only increase the long-term component (significance), but also cause it to become more stable. The purpose of this architectural principle is that if a concept, or a relation between concepts, is encountered multiple times, it should become both more important as an idea in memory, and also harder to forget.

A consequence of the above is that when a link between two related nodes "relaxes", it does not return exactly to its original length, but to one that is slightly shorter. This visualization makes sense if the length of a link corresponds not to the activation or the significance of the relational node but to its weight.

A second consequence is that memories can be built that are literally unforgettable. If the stability of the significance increases beyond some value, the time it takes for the significance to drop back to "zero" can become longer than the lifetime of a system employing Phaeaco's architecture.

9.3.2 Links as associations and as relations

An additional difference between Phaeaco's conceptual network and a traditional Slipnet is that in Phaeaco two nodes can be associated with a link without this link implying a node that represents explicitly a relation.

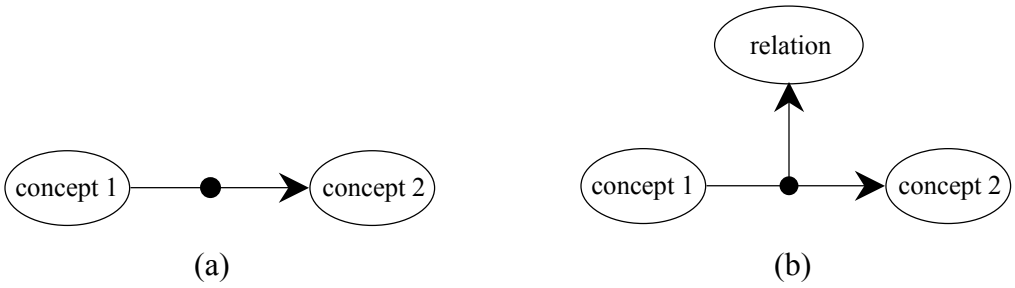


Figure 9.5: An association (a), and a relation (b)

Figure 9.5a depicts a simple association between two concepts without an accompanying explicit relational node. The black dot in the middle of the link connecting the two associated concepts is an entity that can include an activation and a significance. But no reference can be made to this association — in other words, it is an "anonymous relation". In such cases what the system knows can be expressed as: "concept 1 entails concept 2", or: "the two concepts have been seen together", without further elaboration on the nature of their relation.

However, every co-occurrence of the two concepts increases the activation of their association, and thus its overall weight. When the weight exceeds a threshold (which is a parameter of the association), a node representing explicitly the association (now called a "relation") is built, as in Figure 9.5b. At that point, the dot shown on the link in figures of this text represents nothing but a structure that simply records which nodes are connected. The activation and significance values are now both in the relational node, and the structure is the same as in a Slipnet.

Once a relation is built, it cannot be destroyed, i.e., there can be no return to a simple association.[67]

The reason for allowing simple associations besides explicit relations in Phaeaco's architecture is that not every relation appears to be an explicit concept, worthy of a "handle" (i.e., a relational node) by which it can be referenced from (and make references to) other concepts. For instance, most connections between nodes in a core structure (§9.2.1) are simple associations — too mundane to be elevated to the status of a relation that can be *talked about,* and possibly matched against similar relations. However, associations are not condemned to anonymity forever. Repetition can cause the association to strengthen, and eventually to acquire a "label" (a node) that marks it as a relation.

There is also a deeper justification for simple associations. Phaeaco does not model only the most intricate and commendable accomplishments of human cognition. Its domain is vision, and humans are not the only animals that see the world. They are, without doubt, the only animals that can entertain thoughts such as: "A square is a special case of a rhombus" or "Opposite is the opposite of similar". Relations in such thoughts are treated as objects that can be talked about. In Phaeaco's conceptual network (and also in any Slipnet), relations are reified as in Figure 9.5b. But it is doubtful that any animal[68] can reify relations as humans do. If some cognitively complex animals entertain rudimentary representations, it seems plausible to assume that these consist of simple associations, rather than explicit (reified) relations. And because human cognition did not spring out of nonexistence suddenly one day fully equipped with explicit relations, simple associations must also be an integral part of human visual cognition.

---

[67] However, a relation, as well an association, can be permanently forgotten by means of a mechanism described in §9.4.2.2.

[68] With the possible exception of a few laboratory-raised and trained mammals and birds.

9.4 How to remember, and how to forget

Some practical considerations are examined in this section. Specifically, methods are suggested for locating concepts in LTM without resorting to a sequential search, and also for discarding "forgotten" concepts and connections, thus avoiding the eternal expansion of memory to an ever greater size.

9.4.1 Indexical nodes

Some of Phaeaco's architectural features are dictated by the demand that Phaeaco function reasonably well even if implemented in computing systems that do not employ a massively parallel architecture, as the human brain does, but in serial ones, which compensate for the lack of parallelism with computing speed. In LTM the lack of parallelism in the underlying hardware is problematic, because if a new representation must be matched against all existing concepts, it is impractical to examine each of them in sequence, given that pattern-matching and categorization are relatively expensive operations.

This problem is solved by the designation of some nodes as "indexical". An indexical node can be a copy of any relational or numerosity node of a core structure. The task of the indexical nodes is to make it possible to locate relatively quickly a set of candidate concepts for matching with a new representation in the Workspace. The set can include more candidates than necessary, but must not miss the concept that most closely matches the given representation. The indexical nodes stand as a separate layer, or interface between the Workspace and the rest of the LTM. An example will clarify their creation and use.

Suppose the LTM has not yet learned any new concepts (i.e., it contains only the primitive concepts that are built into it, such as "slope", "line segment", etc.), and that the representation of a rectangle — already constructed in the Workspace

— must migrate into the LTM. There are several relational and numerosity nodes in the representation of a rectangle, shown in Figure 9.6.

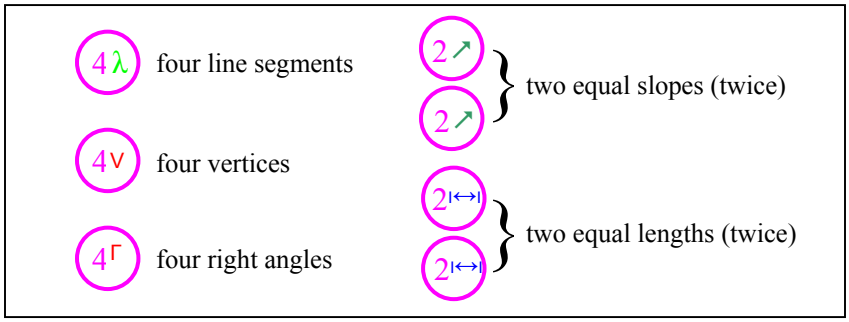


Figure 9.6: Numerosity and relational nodes of a rectangle

A copy of these nodes is made and placed into the "LTM index", which is merely a collection of indexical nodes that stands "between" the Workspace and the conceptual network of the LTM (Figure 9.7).

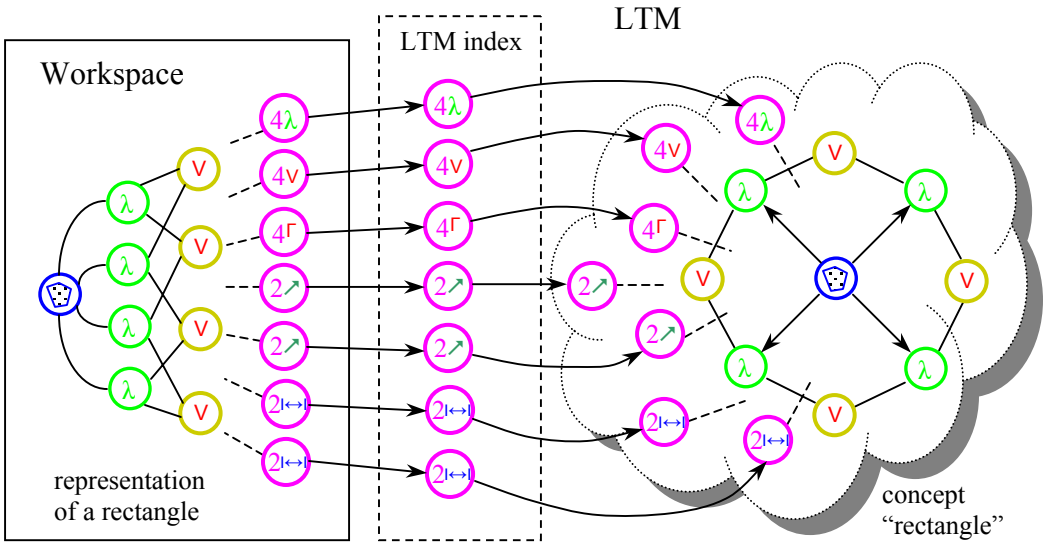


Figure 9.7: The LTM index, an interface between the Workspace and LTM

Any reappearance of a rectangle in the input will activate exactly the same indexical nodes in the LTM index, which in turn will pass their activations to the core of the concept "rectangle" (only a few of the relevant links are shown in Figure 9.7). Suppose now that a trapezoid is seen in the input. Figure 9.8 shows the indexical nodes of a trapezoid.

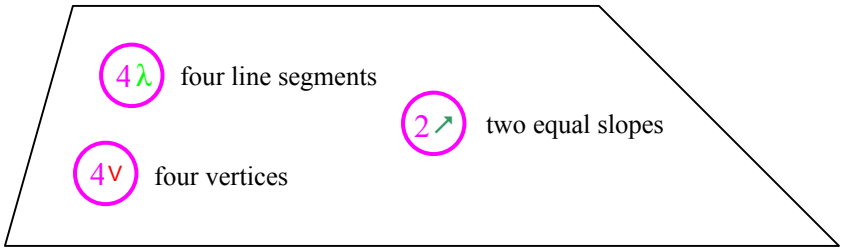


Figure 9.8: Indexical (numerosity and relational) nodes of a trapezoid

This time only a subset of the indexical nodes in the LTM index will receive activation, as Figure 9.9 shows.

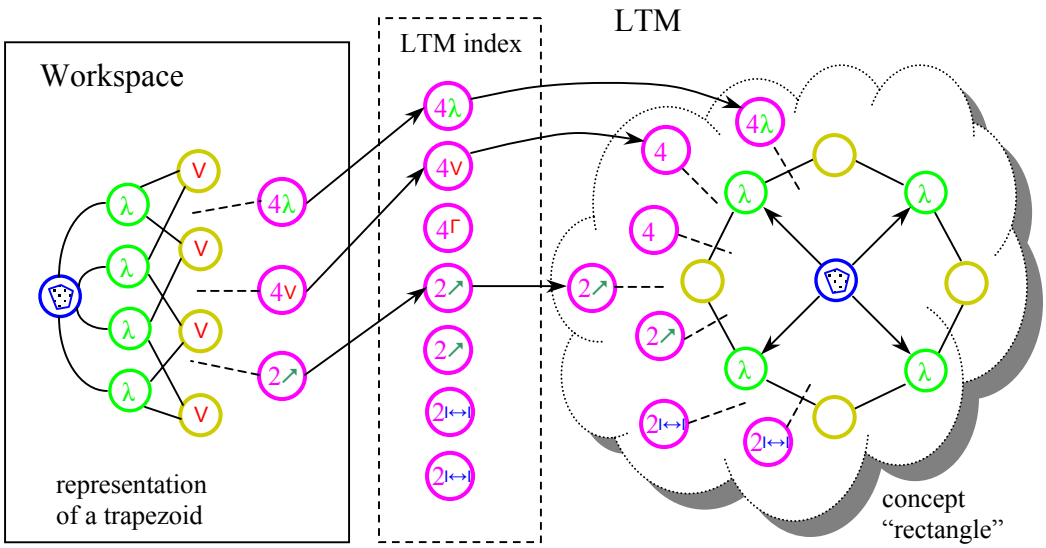


Figure 9.9: A subset of indexical nodes is activated by a trapezoid

Thus, given a trapezoid, the core of the concept "rectangle" will not receive as much activation as it would if a rectangle were present in the input.

Assume now that the pattern of a trapezoid is judged to not match sufficiently well with "rectangle", and so "trapezoid" is also established as a concept in LTM. Given a new trapezoid in the input, according to the previous discussion, both "rectangle" and "trapezoid" will receive the same amount of activation, since all of the indexical nodes of "trapezoid" are also shared by "rectangle". To force "rectangle" to receive less activation given a trapezoid, the inactive indexical nodes of "rectangle" *inhibit* by a small amount the activation at its core.

The purpose of the indexing scheme is to identify not *the* conceptual core with the highest activation, but *a set* of concepts that receive activations beyond some threshold. Once such a set of candidate concepts is established, each member of the set is matched against the given representation to determine the best match.

The nodes of the LTM index are not pre-existing, but are established as new concepts arrive and reside in LTM, bringing their new indexical nodes with them. For example, if a triangle is seen next (continuing the previous example of a rectangle followed by a trapezoid), then indexical nodes such as "three line segments" and "three vertices" will be added to the LTM index.

9.4.2 Forgetting concepts

9.4.2.1 Justification of forgetfulness

An essential difference between the memory of a computer and a cognitive memory is that the former is indiscriminate, or "lossless", storing all information and making it available forever, whereas the latter is selective and prone to forgetting. Although forgetting might at first thought appear as a drawback, in reality it is a mechanism that adds power to memory, rather than detracts from it.

One obvious justification for not storing explicitly all memories forever is the sheer amount of information a biological organism is confronted with in its environment during its lifetime. Given that brains did not appear suddenly at full capacity as we know them in our species, but evolved gradually from very simple neural devices, there was hardly any option but to have the selectivity of memory built into brains from the very beginning. There is more to forgetting, however.

An issue often mentioned in the literature on learning is the problem of overgeneralization in the absence of negative examples (e.g., Berwick, 1986). If the learning system receives both positive and negative examples of a set of elements to be learned, then there is no problem in reaching a description that correctly characterizes the set: positive examples expand the "boundary" of the description of the set toward generalization (see Figure 9.10), and negative examples prevent it from expanding and including elements not in the set (Mitchell, 1978). This is part of what is known as "inductive learning".

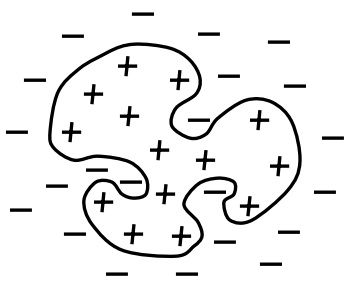


Figure 9.10: With both positive and negative examples, the set can be delineated properly

The problem is that in many learning tasks, negative examples are not available. A child who learns a language, for instance, is seldom (and in some cultures never) informed of which morphosyntactic generalizations are wrong. Most linguistic learning is achieved by exposition of positive examples (see, e.g.,

Brown and Hanlon, 1970; Wexler and Culicover, 1980). But then a question naturally arises: how is overgeneralization avoided? For example:

- "timidity" is the state or quality of being timid,
- "immensity" is the state or quality of being immense,
- "authenticity" is the quality or condition of being authentic,

but

- "electricity" is not the state (or quality or condition) of being electric.

Some "timid learning" mechanisms have been proposed as an answer, such as the "subset principle" (Berwick, 1986), in which an inductive system makes the smallest possible generalization from the given data. However, because inductive learning systems have no understanding of the importance of features, they often make faulty generalizations. For example, when a program called IPP learned about two bombings in India, each of which resulted in two deaths, it generalized that bombings in India always kill two people (in Schank and Leake, 1990).[69]

Computational approaches to inductive learning generally suffer for a deeper reason: they treat all knowledge as eternally present — a consequence of the dimension of time being absent from computer memory. Because the property of being lossless is superficially seen as an advantage, few AI system designers have wondered whether there can be advantages in forgetting some facts. (But see Roger Schank's work on dynamic memory for a notable exception (Schank, 1982)).

Consider a system that encounters only positive examples, but does not store them in memory forever. Instead, each example (each cross in the abstraction of Figure 9.10) stays in memory for some time, but requires confirmation to justify

[69] It is questionable whether IPP had even a rudimentary understanding of "bombing", "India", "death", "to kill", "people", or even "two-ness", for that matter.

its presence, otherwise it gradually fades from memory. Suppose also that the system has made some wrong generalizations (false positive assumptions). This situation is depicted in Figure 9.11, in which examples of false positive assumptions that have not received recent confirmation are shown as crosses with varying degree of intensity in their color.

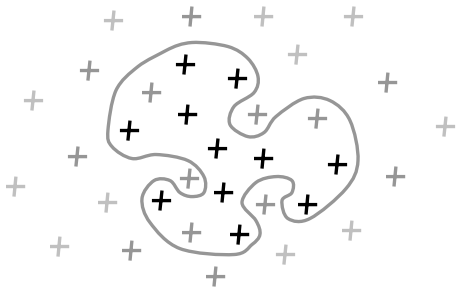


Figure 9.11: Inductive delineation of a set using only positive examples and the time dimension

All dark crosses in Figure 9.11 are assumed to be examples that have been recently activated (confirmed) multiple times. Thus their "weight" remains at a high level. Crosses outside the set are generalizations that were made by the system (false positive assumptions) some time ago, but that were never confirmed, so they have started fading. Some examples in the set are also fading because they were not confirmed recently, but the hope is that they will be confirmed in the future, since they are in reality part of the set. The boundary of the set itself is grayed, because it is not delineated with perfect certainty at any time. Finally, for the system to succeed, some mechanism analogous to Phaeaco's increase of "stability" in the retention of memory is necessary, otherwise all memory will be eventually erased, given enough time.

An additional possible justification for forgetfulness in cognitive memories is the selectivity in what must be remembered. As everyone knows, memories are

not stored photographically (exceptional individuals notwithstanding), but abstractly. The previous paragraphs justifying inductive learning through forgetting, for example, cannot be remembered verbatim, but only abstractly. Similarly, neither the shape that appears in Figure 9.10, nor the locations of the plus and minus signs can be remembered with absolute accuracy by normal individuals. As pointed out in FCCA (Hofstadter, 1995a, p. 212), the greater its conceptual depth, the more important a concept is considered once it is perceived. If analogy-making is truly at the core of cognition, then literal mindedness cannot be an essential component of it, because the best analogies involve the most abstract elements of a situation, or situations. Thus, a deeper understanding implies some degree of forgetting.

Cases of exceptional individuals with so-called "photographic" memory appear to confirm this view. Particularly well-known is the case of the mnemonist "S.", who was able to recall sequences of random letters and digits listed in the form of an array on a page, and could recite them in any order (forward, backward, or diagonally), decades after first seeing them. Nonetheless, the same individual was incapable of understanding the meaning of a phrase with the slightest hint of abstraction, such as "the work went underway normally" (Luria, 1968). The reader of Luria's account of S.'s abilities is left with the impression of being offered a glimpse into the workings of the memory of a modern computer in which the superficial and specific have displaced the deep and essential.

9.4.2.2 Forgetting in Phaeaco

For a concept in Phaeaco's conceptual network to be "remembered" without being present explicitly in the input, its weight must exceed a certain threshold. The (long-term) significance alone is insufficient to exceed this threshold, so that some amount of activation must reach the core node of the concept from its

neighboring nodes. The amount of activation that spreads from one node to another depends on the "length" of the link that connects the nodes (where the length is a function of the weight of the association or relation, as explained in §9.3.2). Hence, the more distant two linked concepts are, the harder it is to remember one given the other. Forgetting is automatically implied by the tendency of links to "relax" (extend) as time goes by in the absence of any reinforcement that would cause them to shrink once more. Thus forgetting in Phaeaco means that, over time, activation cannot reach some concepts.

Even so, that some concepts are unreachable does not imply they are absent from the conceptual network. It is doubtful whether concepts in human memories are ever erased completely, but if they are, it probably happens gracefully, with some synapses devoted to the memory of a concept reassigned to the memory of a different one. But for a computationally implemented memory, it is probably best to include some mechanism to actually delete unreachable nodes. Phaeaco does this implicitly: when its memory is saved permanently on secondary storage, links with weight at its minimum value (the "limiting activation", §9.3.1), are not followed by the algorithm that visits the memory nodes; hence, the nodes that are unreachable (by *any* link) are never saved in secondary storage. When the conceptual network is loaded back into memory, the unreachable nodes are absent from Phaeaco's LTM.

9.5 Conclusion: what does "learning" mean?

The notion of "learning" has been used quite extensively in psychology, AI, and biology, but there is no consensus on what exactly learning *is*. The problem with this concept is that everybody has an intuitive understanding of it — as is the case with "intelligence", for example. Not only is the concept familiar, but everyone

believes they have first-person experiences of learning. As a result, the word "learning" is easy to understand, so it has been used in a variety of instances that could be described more accurately as "memory update". The present section is not an attempt to redefine the meaning of "learning", but to review very briefly the ways and domains in which the term has been used, and to contrast them with the meaning of "learning" as it is used in Phaeaco. Specifically:

- In biology, the notion of *habituation* is usually associated with the more general concept of learning. An organism habituates when it stops responding after repeatedly being exposed to the same stimulus. For example, a spider that does not consider a species of ants edible learns, after the first few encounters, to stop rushing to various parts of its cobweb where such ants keep getting entangled. Organisms as primitive as sea slugs are known to be able to habituate to repeated stimuli (e.g., Arms and Camp, 1988, p. 577). Habituation is simulated in Phaeaco's architecture (though no claim is made that it is also *modeled*) in at least one instance: when an idea for a solution to a BP has been generated and already found to be wrong, it can keep "bugging" the solution-seeking module, but each time with a diminished urgency. More about this will be explained in chapter 11.
- In cognitive psychology, forms of memory and learning that have been explored extensively include classical conditioning, operant conditioning, and priming (Crick, 1994). Of these, the first two can be thought of as forms of association-building in Phaeaco, although the specific mechanisms are not modeled. Priming, however, is the main mechanism in Phaeaco by which the context can have an effect on perception (see §8.2.3, Equation 8.7).
- In AI, there are a variety of ways in which the term "learning" has been used. The general term is "machine learning" (e.g., Mitchell, 1997), under which

several approaches to learning have been considered, including inductive learning (mentioned in §9.4.2.1), decision-tree learning, Bayesian learning, genetic algorithms, and, last but not least, artificial neural networks (ANN's). None of these methods is related to learning in Phaeaco, except possibly ANN's, because some features in Phaeaco's conceptual network resemble those of an ANN — a resemblance that can be misleading. Learning in ANN's deserves some further elaboration.

Some connectionists assume (often tacitly, to avoid engaging in unproductive discussions) that theirs is the only "true learning" in the world of computation. After all, some features in ANN's appear to be closely approximating those of brains: recognition from partial input (the filling-in of missing information), graceful degradation (the network does not crash if some units malfunction), and a machinery seemingly not tailor-made to the problem at hand, but consisting of "a number of units", usually connected in some principled way. But a critical look at the way ANN's learn reveals some disturbing properties.

The following example is typical of ANN behavior. Geoffrey Hinton devised a three-layer network to compute family relationships (Rumelhart, Hinton *et al.*, 1986). (He intended it to be a demonstration of how error back-propagation works in ANN's, but some connectionists have treated it as a genuine theory of psychology.) After some training, the network could answer questions about who is related to a named person in a given way. Steven Pinker offers the following criticism of this ANN, extending it to ANN's in general (Pinker, 1997, p. 130).

> After training the model to reproduce the relationships in a small, made-up family, Hinton called attention to its ability to generalize to new pairs of kin. But in the fine print we learn that the network had to be trained on 100 of the 104 possible pairs in order to generalize

> to the remaining 4. And each of the 100 pairs in the training regime had to be fed into the network 1,500 times (150,000 lessons in all)! Obviously children do not learn family relationships in a manner even remotely like this. The numbers are typical of connectionist networks, because they do not cut to the solution by means of rules but need to have most of the examples pounded into them and merely interpolate between the examples. Every substantially different kind of example must be in the training set, or the network will interpolate spuriously, as in the story of the statisticians on a duck hunt: One shoots a yard too high, the second shoots a yard too low, and the third shouts, "We got him!"

The thousands of repetitions that are necessary for an ANN to "learn" serve as an indication of the distance of ANN's from the methods by which humans learn. Although even a single input presentation is sufficient for Phaeaco to learn something new, and a few repetitions are sufficient for the formation of a concept, these facts will not be used to argue that Phaeaco's learning is more human-like than that of ANN's.

In summary, it is now worth reviewing what "learning" means in Phaeaco:

- Adding concepts: New core structures can be added in the conceptual network and linked appropriately to existing concepts.
- Enriching known concepts: The statistics of featural nodes, as well as the strengths of links within a structure, can be updated, resulting in concepts that reflect in a more informed way the examples that have generated the concept.
- Shortening the "distances" between nodes, making it easier for activation to spread among them, thus reaching faster from one primed concept to another.
- Forgetting: Phaeaco's memory can be made to selectively include only relevant and current information.

CHAPTER TEN

Image Processing

As was discussed in chapter 4, if it were expected that some human "helper" would convert BP's from their original form (e.g., as printed in Bongard's book) to some internal representation (for instance, as in chapter 7) before Phaeaco could process BP's further and solve them, the present approach would be at best unremarkable — perhaps even dishonest, according to §4.1. This chapter explains how Phaeaco's retinal level (§4.3) perceives the pixels given as input, acting in cooperation with the cognitive level. The two levels work in a "pipelined" fashion (Figure 10.1).

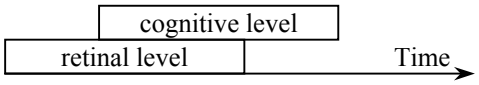


Figure 10.1: Pipelined execution of retinal and cognitive levels

Figure 10.1 indicates that the cognitive level (already reviewed in chapters 7–9) does not wait until the retinal level finishes processing the input entirely, but starts its work as soon as possible (the exact instant is explained in §10.3.13). This implies that Phaeaco's retinal level is not an isolated "module" for image processing, but is intertwined with the perception at the cognitive level. Indeed, the retinal level can to some extent receive *feedback* from the cognitive level, adjusting the values of some low-level parameters appropriately, so that it "sees"

what the cognitive level "wants" it to see (always within some limits, without succumbing to flagrant self-deception).[70]

10.1 The preprocessor

Despite the introductory remarks regarding the intertwined operation of the retinal and cognitive levels, there is one set of procedures that does act as an independent module. This is Phaeaco's visual preprocessor, the purpose of which is to convert any image — even one of photographic quality — into a black-and-white figure that corresponds in a very faithful way to the original one. The term "black-and-white" means that the pixels of the resulting image are exactly as in a BP: either black or white, as shown in Figure 10.2, with no intermediate shades of gray.

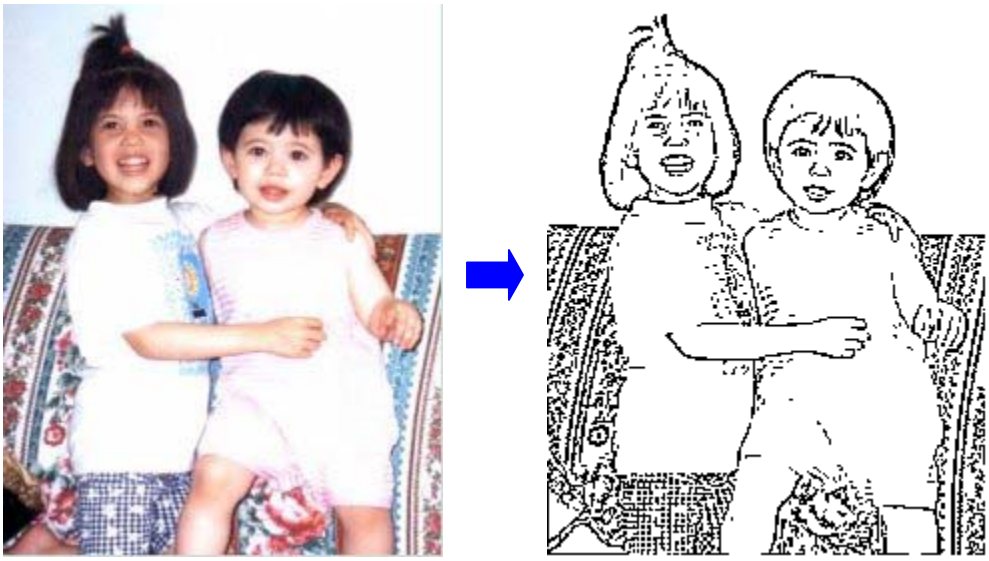


Figure 10.2: Left: original image; Right: black-and-white rendering by the preprocessor

[70] For example, if the context contains many circles, the routine responsible for circle detection (§10.3.12) will become slightly more flexible than usual in accepting a round shape as a circle.

The reason for including a preprocessor as a first step in Phaeaco's operation is the generality of the architecture. Recall that Phaeaco must not be limited to the BP domain, but must be able to handle any visual input. At present Phaeaco cannot understand anything that does not belong to a flat geometric world. But if no attempt were made to design Phaeaco so that it can accept general visual input (of photographic quality), then some simplifying assumptions about the nature of the input would undoubtedly be built deep into its architecture, thus hampering efforts to extend the system in the future.

The preprocessor applies standard image-processing filters to the input pixels to achieve the desired transformation to a black-and-white figure.

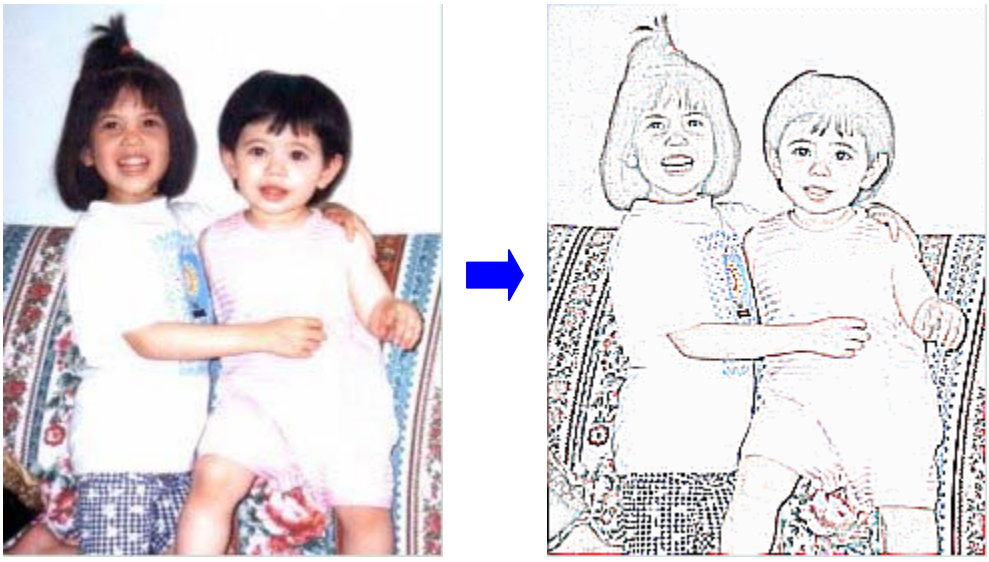


Figure 10.3: First filtered transformation applied on original image (left; result on the right)

For example, Figure 10.3 shows the result of the first such transformation. The filter consists of the 3 × 3 array of integers shown in Table 10.1. To "apply the filter" means that the red, green, and blue components of each pixel are processed independently. For concreteness, focus on the red component, which is

processed as follows: The 3 × 3 array is centered on the pixel to be processed, and the central value of the array (the number 8 at row 2, column 2) is multiplied by the "red" component of that pixel. (Color components are usually in the range [0, 255].) At the same time, the red components of the eight neighbors of the current pixel are multiplied by the integers in the corresponding locations of the 3 × 3 array. The sum of the nine products is added to a so-called "bias factor" of 255. If the total sum falls outside the allowed range [0, 255], it is truncated to either 0 or 255, accordingly. The resulting value is the new "red" component of the current pixel. An identical transformation is applied to the green and blue components, yielding a pixel with modified red, green, and blue components that replaces the original pixel.

-1	-1	-1
-1	8	-1
-1	-1	-1

Table 10.1: Filter for obtaining the "contours" in an image

It is now a simple matter to produce the desired image on the right of Figure 10.2, having obtained the image on the right of Figure 10.3: every pixel sufficiently close to white must become white; otherwise it becomes black. The Euclidean formula for distance can be employed to make the notion of "closeness" between two colors concrete: each pixel is a 3-tuple (red, green, blue), so the color-distance d_c between two points (r_1, g_1, b_1), and (r_2, g_2, b_2) can be computed by the formula: $d_c = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$.

A threshold value t must be used in the above transformation: assuming the distance d_c of each pixel from the color white (255, 255, 255) is computed, then if

$d_c < t$, the pixel must be converted to white, otherwise to black (0, 0, 0). The value of t that yields the best results depends on the quality of the image. For now, Phaeaco uses a fixed threshold, but future extensions must employ some algorithm to derive it from the properties of the image.

10.1.1 Determining the background color

The preprocessor is not invoked *unless* the image is a photograph. How is this determined? Imagine a *pre*-preprocessing routine that counts the colors of the pixels: if the number of colors[71] is more than two, then the preprocessor is invoked and converts the image to one with exactly two colors, as just described. Otherwise (if the colors are exactly two, with whatever exact hue) there is no need for preprocessing.

However, even after a two-colored image has been obtained, one final question remains before the image can be considered as input for a BP: which of the two colors comprises the foreground, and which the background? It is not always possible to answer this question categorically.[72] But for input that can be used in boxes of BP's it is usually possible to decide among the two colors.

Phaeaco uses a heuristic to answer this question: it considers the pixels in a frame around the border (the thickness of the frame is 5% of the width and 5% of the height of the image). The color that is more than 50% in that frame is deemed the "background". (If the heuristic fails, so will Phaeaco in solving the BP.)

Once a decision about the identity of the background color has been made, the pixels are transferred to an internal array of integers, in which **0** stands for background, and **1** for foreground. All further processing uses this internal array.

---

[71] Grayscale counts as "colors", too.

[72] For example, consider a checkered image alternating black and white squares.

10.2 The pipelined process model

The main image-processing stage (following the preprocessing one) is organized in a number of processes, labeled by the letters of the alphabet (A, B, C, etc.), which run in a pipelined fashion, as in Figure 10.4.

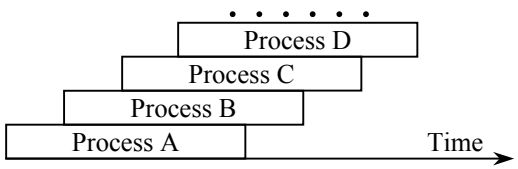


Figure 10.4: Pipelined processes at the retinal level

Each process specializes in some particular aspect of the input: for example, line segments, line intersections, curves, closed regions, angles, circles, dots, etc. Not all letters of the alphabet correspond to a process (currently there are 17 in total, to be discussed in §§10.3.1–10.3.17), nor do the processes run in strict alphabetic order: some might never be initiated, if the input lacks the features on which those processes specialize.

The principle behind this organization of the retinal level is that each process "lurks" in the background, and starts working as soon as data upon which it knows how to operate become available.[73]

A possible criticism of this organization is that the FARG model of codelets could be employed in Phaeaco's image-processing stage. Codelets looking for specific features of the input — some for straight lines, others for intersections, yet others for curves, and so on — appear ideal as a tool for this task.

---

[73] Recall that the same principle was mentioned in the discussion on the pipelining of the retinal and cognitive levels.

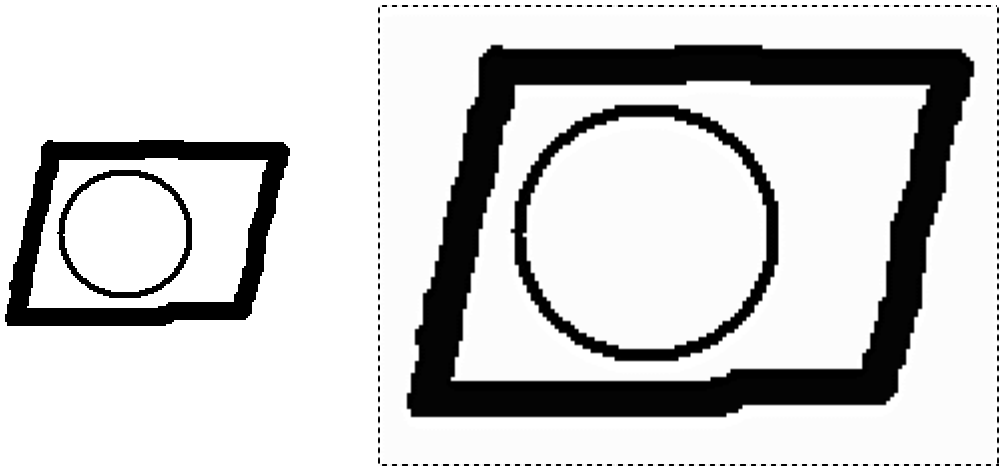
The answer is to be found again in Figure 4.9, which conveys pictorially the philosophy behind Phaeaco's architecture. If Phaeaco were an attempt to simulate the functioning of the *brain,* and in particular the parts of the visual cortex that process complex visual notions ("shape", "dent", "bump", "turning clockwise", etc.), then codelets would be a possible candidate for an architectural framework. But, as Figure 4.9 suggests, the nature of computer hardware, which is fundamentally different from that of neural structures, calls for an approach that is free to utilize the properties of computers at the retinal level and that simultaneously heeds the requirements of the cognitive level.

Nonetheless, Phaeaco's pipelined process model is *inspired* by the way the primate visual cortex is organized into modules. For example, besides the two retinas at the front end of the visual system, the visual cortex of primates includes more than 20 regions (V1, V2, V3, etc., but their labels are not universally accepted), which are devoted to the perception of line segments, slopes, lengths, color, motion, etc. (Posner and Raichle, 1994; Thompson, 1993; Zeki, 1993). There is no one-to-one correspondence between Phaeaco's processes and the visual cortex areas, but there is an abstract similarity in organization. For example, most of the input to area V2 of the cortex is from V1, most of the input to V3 is from V1 and V2, and so on — a progressive funneling of visual information (Thompson, 1993, p. 244). Also, receptor cells in V1 are simple, whereas most of the cells in V2 are of a type called "complex", and more than half of the cells in area V3 are "hypercomplex" (*ibid.,* p. 245). A similar progression in complexity of processing routines and funneling of processed information also characterizes Phaeaco's retinal-level organization, which is explained in detail in the rest of the present chapter.

10.3 The retinal processes

10.3.1 Process A

The first retinal process is independent from almost all other processes[74]. It is also the simplest, testing pixels of the image at random locations, and ignoring them if they are **0** ("white" background pixels in a typical BP box), but inserting them in a queue for further processing (by process B, waiting in the pipeline), if they are **1** ("black" foreground pixels that belong to some object[75]). Suppose the input is as shown in Figure 10.5a.



(a) (b)

Figure 10.5: Input for process A: original (a), and magnified ×2 in a visual box (dashed lines) (b)

Figure 10.5 shows the original input on the left. On the right, the input is shown magnified by a factor of 2 to make individual pixels visible, and is also placed in a visual box (dashed lines). Process A takes a random sample of pixels within the visual box, shown in Figure 10.6.

---

[74] See §10.3.10 for an exception.

[75] The word "object" at the retinal level will be used to refer to a group of connected pixels.

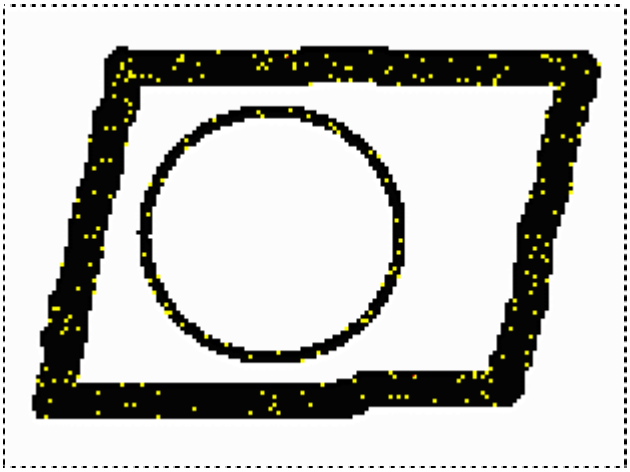


Figure 10.6: Sample of random pixels created by process A

Process A samples the entire visual box uniformly, but only the sampled pixels that are **1** in the input (i.e., not part of the background) are shown in Figure 10.6. The sampling density is a "retinal parameter" of this process. The term "retinal parameter" will be used to denote a parameter that is not hardwired directly into the programming code (it is specified in a file of parameters), but is independent of anything that happens at the higher, cognitive level.

Traditionally in image-processing, a rectangular image is scanned starting at its top left corner and proceeding in a left-to-right, top-to-bottom fashion to its bottom right corner. The reasoning is that all pixels of the image would have to be seen sooner or later, so it is best to scan them in a systematic manner. There are two reasons for not following the traditional method in Phaeaco.

The first is efficiency. Although eventually Phaeaco will process all input pixels that are **1**, the processes in the pipeline do not need to wait until all pixels are seen by process A before reaching their first conclusions. Occasionally this might lead to a situation that appears more like "jumping to conclusions", rather than drawing solid and irrefutable results from the given input. But this is

compatible with the philosophy of Phaeaco's architecture. If any spurious, wrong conclusions are drawn initially, they will be corrected later by other, more informed retinal processes. Figure 10.7 explains further the issue of efficiency gain by offering a preview of the results of retinal processes that will be discussed soon, following process A.

(a) (b)

Figure 10.7: Traditional sequential processing (a), vs. random processing in Phaeaco (b)

The image in Figure 10.7a can be the partial result of a sequential approach that, like Phaeaco, does not wait for the entire image to be scanned before identifying, for example, line segments. Figure 10.7b has the same amount of "ink" as Figure 10.7a, but the "ink" is spread randomly over the figure. In other words, the sums of the lengths of the line segments in the two drawings are identical. Nonetheless, Figure 10.7b has a head start in identifying what the input is, because some collinear segments can be extended (instantly, from their equations) to form longer lines; some of these longer lines can be projected and found to meet at points that are **1** in the input, and therefore possible intersection points of input lines; and finally, the end-points of some line segments can be seen as belonging to a circle, so the suspected circle can be created instantly, and further confirmed by random sampling of points generated from its equation. Nothing of this is possible in Figure 10.7a, because no algorithm can "guess what comes next" in a sequential, top-to-bottom scanning of the input.

A second reason for Phaeaco's random sampling of pixels is that in this way randomness becomes woven into the fabric of Phaeaco's architecture. Cognitive systems, unlike conventional programs, do not always arrive at exactly the same conclusions given identical inputs. Phaeaco's random collection of line segments, as shown in Figure 10.7b, will be different the next time the same input is presented. Although at a sufficiently high level the representation of the input will be identical ("a parallelogram with a circle inside"), the details at a lower level will differ. This might not appear advantageous in a seemingly unambiguous figure such as the one examined above, but randomness is at the basis of every system that can, under more complex conditions, act unpredictably in an unpredictable world. If humans always acted in a predictable way, they would hardly feel creative, or be able to justify their feeling of "free will". This does not mean that the mere existence of randomness suffices to explain creativity and the feeling of free will, only that its absence precludes them (Mitchell, 1990).

Process A does not examine random pixels within the visual box indefinitely. It "self-regulates" its rate of pixel examination, gradually diminishing it, until the rate drops to zero. It can do this because it monitors the ratio of "hits" (pixels that are **1**, and thus belong to some object, and have not been hit before) to the total number of pixels examined. The smaller this ratio is, the less time A allocates for itself to run. (The remaining time is distributed among other processes.) Eventually the ratio becomes so small that A "knows" it is time to retire.

Occasionally, an isolated pixel can be missed by process A. The probability for this is quite low, and it is practically zero for a group of four or more pixels.

10.3.2 Process B

The task of this process is to replace the number **1** of each pixel of an object with another, usually larger integer number, called its "altitude". The altitude must be

approximately proportional to the distance of the pixel from the closest border of the object. The more "internal" the pixel is, the higher its altitude must be.

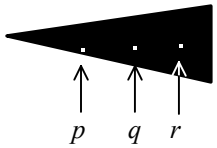


Figure 10.8: Pixels at different "depths" in a figure

Figure 10.8 shows three pixels, *p, q,* and *r,* of an object. Pixel *p* is close to the border of the object, so its altitude must be smaller than the other two; pixel *r* must have the highest altitude, not only among *p, q,* and *r,* but also probably among all pixels belonging to this object.

The purpose of process B is to prepare the ground for the next process, C, which identifies pixels that belong to the *endoskeleton* of the object (§7.4.9).

The task of processes B and C (combined) is known as "thinning" in the image-processing literature. Traditionally, thinning is achieved by successively "peeling off" pixels that belong to the border of the object, layer after layer: eliminating one layer of border pixels exposes the next layer of pixels, which now become the new border, and so on. When no more layers can be peeled off (because doing so would eliminate the object), the last remaining pixels constitute (approximately) the endoskeleton of the object. The problem with this thinning technique is that all pixels must be available before the procedure starts. But this conflicts with Phaeaco's fundamental principle that work must start as soon as input becomes available, even with only partial information, and without waiting for prior stages to be completed.

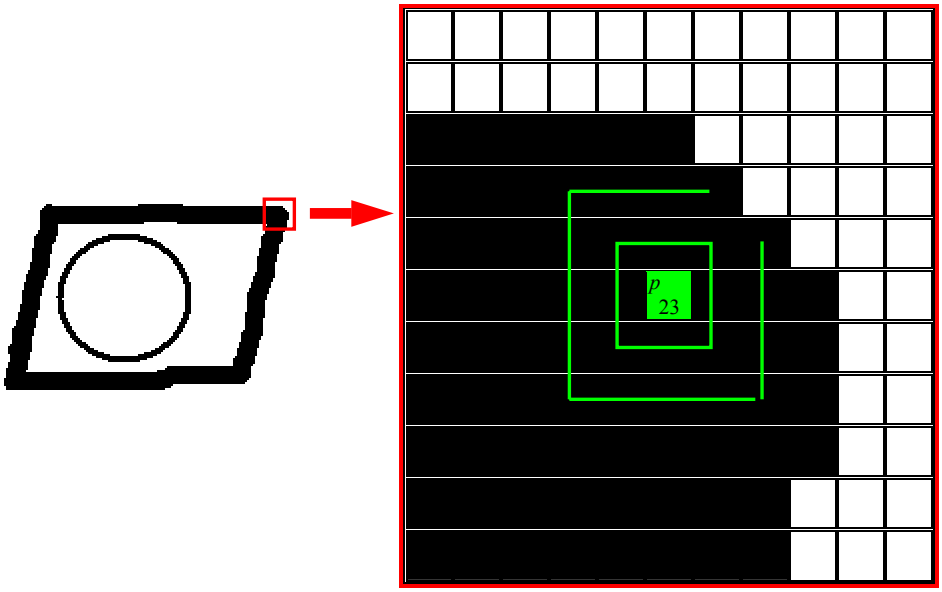


Figure 10.9: A piece of input (magnified on the right) and concentric squares around a pixel

Process B accomplishes the task of assigning an altitude to each pixel it retrieves from the queue of pixels created by process A as follows: given a pixel *p,* it examines the neighbors of *p* in successive "concentric squares", and counts the number of neighbors that are not **0** (Figure 10.9). The altitude of *p* is the total number of its neighbors that are not **0**. For example, the first square centered at *p* comprises its eight immediately neighboring pixels, so the number of those pixels that are not **0** is counted (call it *n*); the next square, concentric to the previous one, is larger and is made of pixels that surround the previous square; the number of pixels along the perimeter of the new square that are not **0** is added to *n*; and so on. The increase in the size of the concentric squares stops as soon as a pixel is found that is **0**. The final value of *n* becomes the altitude of pixel *p*.

Figure 10.9 shows the original input on the left, and a small piece from the upper-right corner of the parallelogram magnified on the right. The figure shows how process B would calculate the altitude of pixel *p* at the center of the

magnified region. Two concentric squares centered at p are shown: the innermost square consists of the eight immediate neighbors, and the outermost one includes 15 pixels that are not **0** (black), and one which is **0** (white, at the upper-right corner). Since a **0**-pixel is found, the algorithm stops and does not examine larger concentric squares. The total count of non-**0** (black) pixels in the two squares is $8+15=23,$ and that is the altitude of p.

Process B helps process A to concentrate on non-**0** pixels by asking A to examine the eight neighbors of the pixel under B's focus, instead of other, random pixels. In this way, B usually manages to compute the altitude of a pixel without waiting for A to see *all* the pixels of the input image, in A's random fashion.

Process B ends when there are no more pixels left in the queue. Like A, it self-regulates the time allocated for it to run, by monitoring the length of the queue: the shorter the queue, the more certain B becomes that it is approaching its time to stop.

10.3.3 Process C

This is a companion of process B, completing the task of identifying the pixels that belong to the endoskeleton of an object (or "median pixels", as they are called in the literature). Process C achieves this by examining each pixel p for which process B computed its altitude, as well as the altitude of its eight closest neighbors. Pixel p belongs to the endoskeleton if there are no more than two closest neighbors with altitudes strictly greater than the altitude of p.

Note that by the above definition, isolated pixels, or pixels that are lined up forming a line of width 1, belong to the endoskeleton.

Figure 10.10 shows an example of an endoskeleton pixel on the left, surrounded by its eight neighbors, and a counterexample on the right. The altitude of each pixel is shown, printed in bold type if it is greater than the altitude of the

central pixel. On the left, only two neighbors are "taller", so the central pixel is deemed part of the endoskeleton. In contrast, on the right there are three taller pixels, so the central pixel is eliminated from the endoskeleton.

Figure 10.10: Example of endoskeleton pixel on the left, and counter-example on the right

Process C usually ends simultaneously with process B. It is also responsible for sensing *filled* regions: pixels that belong to the endoskeleton and have an altitude that exceeds a certain threshold obviously belong to a filled region. A small sample of these pixels (selected at random, because not all are required) are sent to process F (§10.3.6) for further processing.

10.3.4 Process D

Process D is a detector of line segments. The pixels that are identified by process C as belonging to the endoskeleton are placed in a queue, which is read by process D. The problem that this process must solve is: how can it be determined as soon as possible that some pixels are approximately collinear, given that they are not all immediately available?

For example, consider Figure 10.11: On the left, very few pixels have become available, so it is not clear at all where the line segments are. In the middle, more pixels have arrived, making the line segments barely discernible. Finally, on the right there are enough pixels to let *our* visual system see the lines. But how can this be done algorithmically?

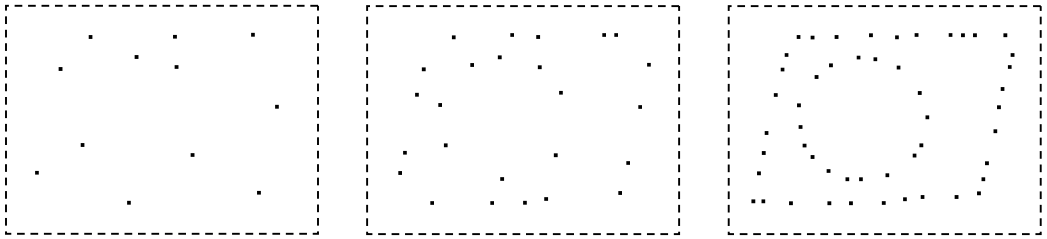


Figure 10.11: Successive stages in the accumulation of endoskeleton pixels

Following the principle that data must be used as soon as they become available, Phaeaco starts forming hypotheses about line segments, even with as few pixels as those in Figure 10.11a. Most of these hypotheses will be wrong ("false positives"). But it does not matter. As explained in §9.4.2.1, if these initial "overgeneralizations" do not receive enough reinforcement in the future, they will decay as time goes by, whereas correct generalizations will endure. Consequently, a *line-segment detector* is a line segment together with an activation value. Indeed, detectors of all kinds of retinal primitives possess activation in Phaeaco. If the data are still at the early stage of Figure 10.11a, some tentative weakly activated detectors will be created (Figure 10.12).

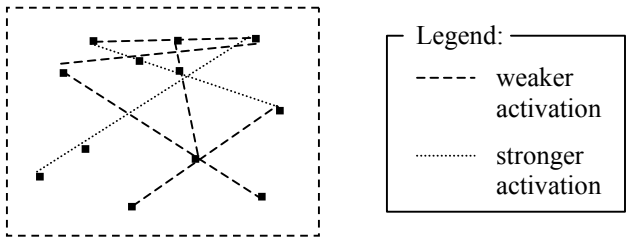


Figure 10.12: Initial tentative line-segment detectors given a few data points

Note that essentially all collinearities of three or more points have been marked with a detector in Figure 10.12. The method of least squares is used to fit

points to lines, and the activation value of each detector depends both on how many points participate in it, and how well the points fit. Only one of these detectors is a "real" one, i.e., one that will become stronger and survive until the end (the nearly horizontal one at the top); but process D does not know this yet.

Suppose now that more points have arrived, as in Figure 10.13. Some of the early detectors will probably receive one or two more spurious points, but their activation will decay more than they will be reinforced. Also, a few more spurious detectors might form. But, simultaneously, the "real" ones will start appearing.

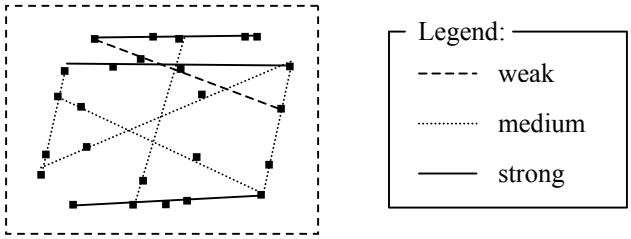


Figure 10.13: More data points keep the evolution of detectors in progress

Note that several of the early detectors of Figure 10.12 have disappeared in Figure 10.13, because they were not fed with more data points and "died". Thus, the most fit detectors survive at the expense of those that do not explain sufficiently well the data. A subtle aspect of this algorithm is that once a data point has been assigned to a relatively strong detector, it is considered "taken" by that detector, so it cannot increase the activation of any other detector.

Overall, the procedure is strongly reminiscent of the categorization algorithm (§8.3.2) employed at the cognitive level: the notion of a "group of objects" is replaced here by a "line-segment detector made of points", and the criterion for "belonging to a group" is "being close to a detector". Chronologically, the

algorithm of process D was designed first, whereas the cognitive categorization algorithm was formed later as an elaboration of the present one.

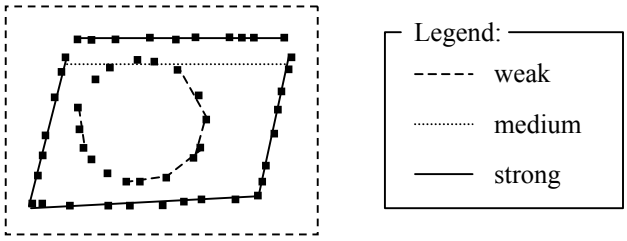


Figure 10.14: Most of the survivors are the desired, "real" detectors

In Figure 10.14, the desired detectors that form the sides of the parallelogram are among the survivors. Also, the tiny detectors that first will form line strings (§10.3.7), then a curve (§10.3.8), and eventually a circle (§10.3.12), have started appearing. In the end, all "true" detectors will be reinforced with more points and deemed *the* line segments of the image, whereas all "false" detectors will decay and die.

Process D is also responsible for the detection of lines that belong to the *exoskeleton* of an object, if the thickness of the object is sufficient to form a filled region. The same procedures that were explained in this subsection up to this point (dealing with pixels of the endoskeleton, which are generated by process C), are also applied to pixels of the exoskeleton, which are generated by process F (§10.3.6).

Process D ends when there are no more pixels to examine in the queues filled by processes C and F. Like all other processes, it self-regulates its time allotment, slowing down when it senses that its supplier queues are drained of pixels.

10.3.5 Process E

This process is concerned with the detection of intersections of line segments found by process D. As before, intersection detectors have activations, but in this case they depend entirely on the activations of the line segments that form the intersection. Thus, this is not an activation that decays autonomously in time, but is instead a function of the activations of the component line-segment detectors. This is necessary, because intersections do not receive reinforcement from data points independently from process D.

The main challenge for process E is to guess correctly where the actual intersections are, since process D initially can generate any number of spurious line segments, and therefore spurious intersections as well. To make a guess, process E examines a small rectangular area around the purported intersection point: if all pixels in the rectangle are **0**, the intersection is ignored (Figure 10.15).

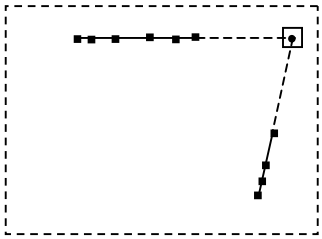


Figure 10.15: Guessing an intersection and examining its neighborhood

Once an intersection is identified as possible, the line segments that form it are extended up to the point of their intersection. (The dashed lines in Figure 10.15 depict the extension of the line segments.) Occasionally, however, the lines intersect at a point that coincidentally is close to a non-**0** pixel, whereas the line segments themselves do not reach that point, as in Figure 10.16. For this reason, a

sparse sample of the pixels comprising the extensions of the line segments is also examined, verifying that the pixels are indeed non-**0**, i.e., part of the image.

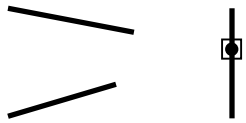


Figure 10.16: The lines on the left, if extended, coincidentally intersect on the line on the right

Process E also records the type of the intersection, i.e., whether it is a vertex, touch, or cross point (§7.4.2; K-points are discussed later, in process K), and stores the information into the structure of the intersection detector.

10.3.6 Process F

Process F is concerned with filled ("thick black") objects. Specifically, it identifies pixels that lie at the border of a filled region and sends them to process D, which thus finds line segments that form the exoskeleton of the filled object.

As was mentioned in §10.3.3, some of the pixels with an altitude greater than a fixed threshold (a retinal parameter) are sent to process F for further treatment. Figure 10.17 shows what process F does with such pixels.

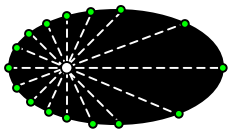


Figure 10.17: Generation of "rays" from a source pixel, searching for border pixels

As suggested by Figure 10.17, process F treats the pixels it receives as sources of a number of "rays" (16 in total), along which it searches for border pixels. A pixel is at the border if three or more of its eight surrounding neighbors are **0**.

Process F repeats this algorithm for all source pixels it receives from process C. A small number of sources generates a large number of border pixels, so only very few source pixels are required by process F.

10.3.7 Process M

The shape of the letter "M" is a mnemonic for the task of this process, which looks for line strings (§7.4.4) among the line segments created by process D: hence its out-of-order alphabetic position.[76] Note that the line strings that this process discovers do not automatically form corresponding nodes at the cognitive level, as described in §7.4.4; these are *retinal-level* line strings, some of which might be precursors of curves, identified as such by process G (§10.3.8).

Process M "wakes up" as soon as an intersection is detected, and tries to make longer the line strings created by intersections. The main issue for process M is that even a simple object with a few intersections usually contains more than one possibility for parsing it into a collection of line strings. For example, in Figure 10.18 the strings ABCEF, ABDEF, ABCEDB, ECBDEF, EDBCEF, and several others, are all possible. (For an extreme case, see Figure 7.31.)

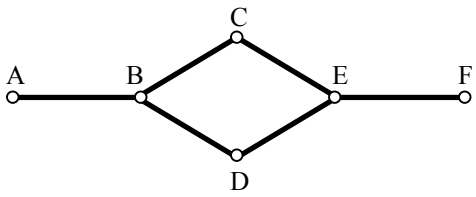


Figure 10.18: Simple input with ambiguous parsing in line strings

If the input contains more than one possibility for a line string, process M will come up with one particular "parsing" of the input. Since the order of line-

[76] Process M must be described before process G, for which it is a prerequisite, because G examines whether line strings form curves (next subsection).

segment identification by process D is random, except that longer lines are usually seen first, the line strings that process M identifies are also random, but biased towards those consisting of longer lines. A subsequent run with the same input is not guaranteed to parse the input into exactly the same line strings, but will be biased toward seeing line strings consisting of the longest lines, and with the largest number of them.

In its attempt for maximization, process M frequently joins small line strings end-to-end, forming larger ones, and eliminating the "subsumed" smaller strings. It also notices when line strings form closed loops.

The decrease of activity in processes D and E signals also a corresponding decrease of activity in process M.

10.3.8 Process G

This is a curve-detector process. As was noted before (in process M), the curves detected by this process do not automatically form corresponding nodes at the cognitive level. After all, some of these curves will be further recognized (by process O, §10.3.12) as circles or ellipses, and therefore will lose their status as "mere curves". Thus it is better to describe them as "curve detectors".[77]

Process G directs its attention to line strings created by process M, becoming particularly interested in strings that are made of several relatively short lines, meeting end-to-end at rather wide (obtuse) angles. The better a line string satisfies these criteria, the more effort process G exerts trying to see it as a curve.

The algorithm by which process G determines whether a line string is a curve examines the intersection points of the line string, and tries to discern whether the

[77] This is a general remark: all elements identified at the retinal level are "detectors", suggesting to the cognitive level what might exist, but not forcing a rigid interpretation of the input. It is the cognitive level that ultimately decides, considering additional context, what exists in the input.

object at those points is curved or forms an angle. It is possible that only part of the line string is curved, with the rest consisting of straight lines; in this case, the algorithm detects a curve only in the part of the line string that appears curved.

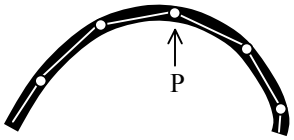


Figure 10.19: Actual curved object and its approximation by a line string

Figure 10.19 shows a curved object (black region) and a superimposed line string that has been detected by process M, approximating the object.[78] Consider point P, shown in the figure. Process G examines how close the endoskeleton pixels are to the line string segments in the vicinity of P (Figure 10.20).

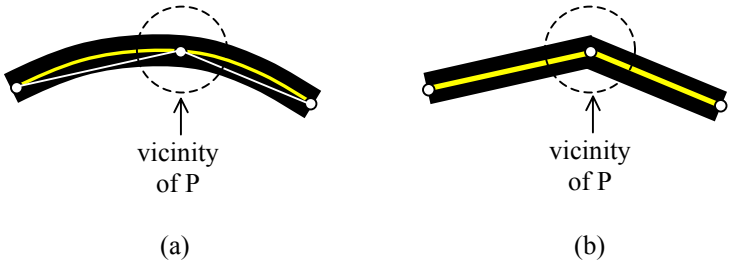


Figure 10.20: Closeness of endoskeleton pixels to the tangent line: curve (a); no curve (b)

Figure 10.20a is a magnification of Figure 10.19 at P. The two segments of the line string that meet at P clearly deviate from the endoskeleton pixels in the vicinity of P. In contrast, in Figure 10.20b the points of the two line segments and the endoskeleton pixels essentially coincide.

[78] In reality the line string is not so perfect: some of the constituent line segments cross or touch each other, rather than meeting exactly at the intersection point as shown in the figure.

Once a piece of a line string has been identified as a curve, there are two steps that must occur before the operation is considered complete: the line string must be replaced, either partially or completely, by the curve; and some way must be established to represent the curve. This is not trivial. The few points of the line string that have been found to belong to the curve are not sufficient to represent the entire curve. A representation is needed that can produce any point along the curve in a continuous manner, so that, for example, the curve can be traced along its length, or so that its curvature can be computed at any desired point, and so on.

The first step is easy: either the line string is deleted, or what is left of it (after the curve has been extracted) remains at the retinal level. For the representation of curves, Phaeaco uses parametric cubic b-splines, explained below.

Suppose the points of the line string from which the curve was derived are P_0, P_1, …, P_n (i.e., a total of $n + 1$ points). Then any four consecutive points (e.g., P_0, P_1, P_2, P_3) define a cubic polynomial $a_3x^3 + a_2x^2 + a_1x + a_0$ in a unique way. If we consider only three points (e.g., P_0, P_1, P_2), then there is a family of cubic polynomials that pass through the three points that depends on a single parameter (the family has one degree of freedom). If we consider only two points (e.g., P_0, P_1), then the family of cubic polynomials that pass through them has two degrees of freedom. Thus we can fit a cubic polynomial between any two consecutive points, but it will be "too free" (indeterminate). To restrict its freedom, we can demand that any two consecutive polynomials (e.g., the one that passes through points P_0 and P_1, and the one that passes through points P_1 and P_2) must be "smooth" at their common point (not form an angle at P_1, i.e., agree on their first derivative), and also agree on their curvature (i.e., agree on their second derivative). These two conditions eliminate the two degrees of freedom. If instead

of a polynomial in standard form we derive parametric equations, i.e., a pair of polynomials $[x(t), y(t)]$, we end up with the formulas in Equation 10.1.

$$S_i(t) = [a_{3i}t^3 + a_{2i}t^2 + a_{1i}t + a_{0i}, \quad b_{3i}t^3 + b_{2i}t^2 + b_{1i}t + b_{0i}], \quad i = 0,\ldots,n$$

Equation 10.1: Parametric cubic b-splines for a curve with $n + 1$ points

Equation 10.1 describes only the form of the formulas, but does not reveal the method by which the coefficients a_{ji} and b_{ji}, $j = 0, ..., 3$, $i = 0, ..., n$, can be computed. The derivation of these coefficients and an algorithm for computing their values from a set of $n + 1$ points are given in Appendix B.

10.3.9 Process H

This role of this process is minor: it performs occasional "corrections" on the output of process D, the line-detecting process. Due to noise in the manner by which pixels form lines, sometimes two or more distinct line detectors identify essentially the same line segment (Figure 10.21).

Figure 10.21: Three line detectors for what should be seen as a single line segment

The example shown in Figure 10.21 is extreme: three line detectors have been created on what should be seen as a single (probably hand-drawn) straight line. In this case, process H will unify all three detectors into a single one, with which it will replace the three original detectors, erasing them from the memory of the retinal level. The algorithm works equally well with lines of either the exoskeleton or the endoskeleton.

10.3.10 Process i

Process i detects isolated "dots" in the input.[79] A dot is anything too small to be assigned an explicit shape, even after applying the magnification algorithm (§10.3.17). The threshold below which something is considered too small to be given a shape is a retinal parameter.[80]

Process D, the creator of line detectors, is also the process that "suspects" the existence of dots: if there is an isolated group of a few pixels, process D fails to "explain" them with a straight line, but it notices their existence, and summons process i to verify the presence of a dot.

The algorithm process i uses to identify dots is trivial. The only subtle point here is that if a fair number of very small dots (e.g., isolated pixels) has been identified, then this process increases the time allotted to process A, to make sure no individual pixel will be missed. The reasoning is that if several dots have already been seen, the odds are that there are more of them in the input.

10.3.11 Process K

Another simple process, again with a mnemonic name, K is assigned the task of detecting K-points (§7.4.2). Its functioning parallels that of process H, except that instead of unifying line detectors it unifies intersection detectors that appear to be too close to each other, replacing them with a single K-point detector.

10.3.12 Process O

Two distinct tasks are undertaken by process O: one is to detect the presence of closed regions and compute their area; the other is to identify circles or ellipses among objects already detected as curves.

---

[79] Hence its name in lowercase — the dot of the "i" serving as a mnemonic.

[80] A radius of 5 pixels works well at medium screen resolutions.

To accomplish the first task, process O selects randomly a line string or a curve. If the object selected is already known to be "closed" (by processes M or G), there is nothing further to detect. Otherwise, it is still possible that the object defines a closed region, but that processes M and/or have G failed to realize this fact (because their algorithms do not specialize in this task, identifying closure only incidentally). Process O performs a more thorough investigation. In addition, it detects near-closure, or any degree of closure, using an algorithm already described in §7.4.7 and depicted in Figure 7.37, which is copied below as Figure 10.22: the ratio of "escaping" rays to the total number of rays is a first estimate of closure, and repetition of this procedure improves the accuracy of the estimate.

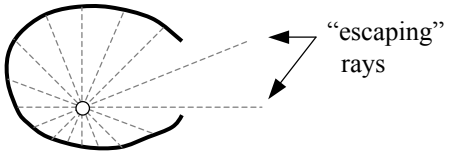


Figure 10.22: Algorithm for computing the closure (or "openness") of a region

In the case of a completely[81] closed region, the algorithm for computing area is triggered, whereas any other region (whether closed or not) triggers the algorithm for computing its convex hull. These will be described in §10.3.18.

The second task of process O involves the identification of the only shapes that Phaeaco is hardwired to recognize: an ellipse in general (in any orientation), and a circle in particular. It could be argued that Phaeaco should be able to learn the shape of an ellipse or circle relying on primitives, just as it does for "triangle". However, the hardwired solution was preferred as a shortcut. The formulas for ellipse and circle identification from a set of points are given in Appendix B.

[81] "Completely" within the certainty allowed by the described approximating procedure.

10.3.13 Process P

P stands for "perception". This is the process that serves as an interface between Phaeaco's retinal and cognitive levels. Process P remains alert as long as activity takes place at the retinal level, noticing the appearance of any "detector" (of lines, intersections, dots, curves, regions, or any detectable object). Its function is to select a detector (probabilistically, according to the activation of the detector) and generate the corresponding cognitive-level node that represents it. Nodes are not merely generated and given to the cognitive level, but placed in the same structure, if they belong together. For example, two line segments that belong to a rectangle will cause the creation of two λ-nodes, and process P will make sure to place them under the same object node (which it will also create).

10.3.14 Process R

This is a cognitive-level process, and its task is to select and run codelets from the Coderack. That it shares the time with all other processes of the retinal level means that initially the cognitive level occupies only a very small time slice in the overall processing of the input. But as the other retinal processes "die" one after another (because there is nothing left in their input on which to work), process R is allotted more and more of the available time, until eventually it is nearly the only process that keeps working.

10.3.15 Process Q

Q is another cognitive-level process, concerned with "quantity" (of anything). For example, if there are several dots in the input, and representational nodes for them are being created by process P, this process will notice that there are "many dots" in the representation. The larger the number, the more probable it is that Q will notice it (usually a quantity above five yields a very high probability that it will be

noticed). What happens after noticing the number depends on the type of the node that appears many times. One course of action (common in all cases) is to slow down, and eventually cease, the generation of *individual* nodes representing input entities, allowing only their numerosity node to be updated. An example of what else can happen with large quantities is given in the next subsection.

10.3.16 Process S

This is a retinal process concerned with the "shrinking" of objects under certain conditions. For instance, process Q might have noticed that an object includes a large number of short lines, as, for example, in Figure 10.23.

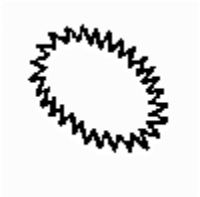


Figure 10.23: Object with many small lines

The representation of the object in Figure 10.23 will initially include several short line segments. But when their number keeps increasing, process Q will notice this, and besides curbing the proliferation of λ-nodes in the representation, it will also signal for process S to apply an algorithm that "shrinks" the object, attempting to perceive its shape by performing the equivalent of looking at it from a distance. The algorithm works as described below.

If an object is shrunk by a naïve method (e.g., turning a pixel in the smaller version to **1** if any of the pixels in a corresponding $n \times n$ neighborhood of the original larger object is **1**), the result is a failure, because all the imperfections of the original object are transferred to the miniaturized one. The purpose of shrinking is to eliminate the imperfections, smoothing out the object as much as

possible. To this end, each pixel in the miniaturized version, instead of being assigned **1** ("black") or **0** ("white"), can acquire intermediate ("gray") values in the range [0, 1], according to how many black pixels exist in the corresponding $n \times n$ neighborhood of the original object.

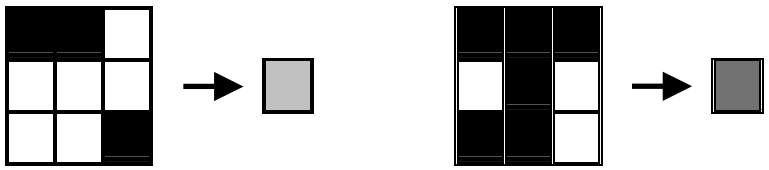


Figure 10.24: Conversion of a neighborhood of pixels into a single pixel with "gray" value

Figure 10.24 shows two examples of 3 × 3 neighborhoods of the original object that are converted to a gray pixel. The example on the left has few black pixels, hence a lighter gray value; whereas the example on the right has more black pixels, hence a darker gray value. The result is shown in Figure 10.25.

Figure 10.25: Intermediate step in the shrinking of an object

Finally, the darkest gray pixels of the intermediate fuzzy object can be converted to black (**1**) (Figure 10.26).

Figure 10.26: Final step in the shrinking of an object

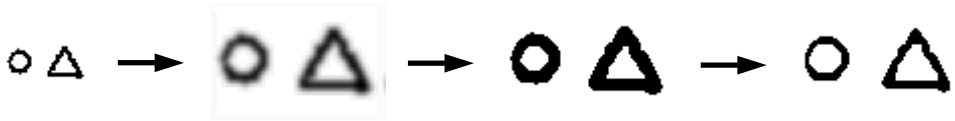
The result is an object on which the entire set of retinal processes described so far can be applied, in order to recognize its shape. Naturally, the representation of the original object in Figure 10.23 will not be merely "an ellipse", but "an ellipse

made of many short lines". Phaeaco cannot reach the description "wiggly line" at present, because it lacks the image-processing acuity to see the way the short lines are arranged with respect to each other.

There are two important parameters in the above algorithm: one is the factor by which the original object is shrunk, and the other is the "fuzziness" factor, i.e., the size of each $n \times n$ neighborhood that gets converted to a gray pixel. In practice a shrinking factor of 0.5 and a fuzziness factor of 5×5 usually yield acceptable results.

10.3.17 Process Z

Z is a retinal process that enlarges ("zooms in") small objects, applying the magnification algorithm announced in §5.1.5. The algorithm is the same as the one described in process S. The only difference is in the value of the parameters: the shrinking factor now becomes a "zooming factor" with a value of 2, and 3×3 pixel neighborhoods of the original small object are converted to fuzzier neighborhoods twice their size (6×6). Figure 5.14 shows an example of the magnification process, reproduced below as Figure 10.27.



(a) (b) (c) (d)

Figure 10.27: Successive steps in zooming small objects

The steps shown in Figure 10.27 are: original input (a); "fuzzy magnification" (b), as described above; turning gray pixels to black or white using a suitable threshold (c); and a moderate degree of "thinning" (d).

10.3.18 Other image-processing functions

In addition to the retinal processes, there are many other algorithms for deriving pixel-based features of the input that can be called when codelets at the cognitive level examine various facets of the representation. The algorithms for deriving the convex hull and computing the area of an object are discussed below.

Procedures described in the image-processing literature for computing the convex hull of a set of points usually assume that all points are available before the procedure starts. But Phaeaco has to perform all tasks incrementally, not knowing or anticipating when the input will be completely processed.

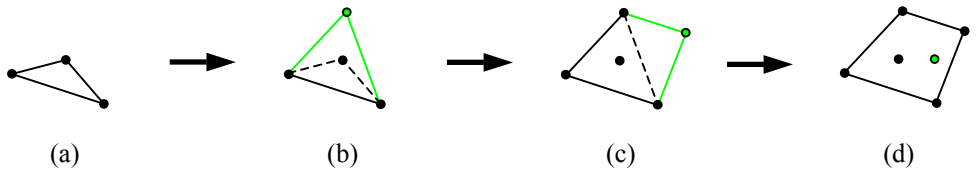


Figure 10.28: Successive stages in the derivation of convex hull

Figure 10.28 shows four successive steps in a convex hull construction. The point added at step (b) causes the deletion of two edges and one point that previously belonged to the convex hull (dashed lines). At step (c) the deletion of a single edge takes place. Finally, at step (d) the new point is ignored.

Last but not least, the computation of the area of an object depends on whether the object is convex or concave. If it is convex, the area of its convex hull is computed by a simple triangularization of the object. If the object is concave, a Monte Carlo method is used: a large number of random points in a box surrounding the object are generated, and the area is found by dividing the number of points that are located in the object by the number of points that are generated in the visual box.

CHAPTER ELEVEN

Putting the Pieces Together

11.1 How BP's are solved

Given a BP, Phaeaco constructs a visual pattern from the six boxes of the left side, and another visual pattern from the six boxes of the right side. It then compares the patterns, attempting to spot some difference between them. This is Phaeaco's method of solving BP's, in a nutshell.

In §8.3 it was explained that a visual pattern is formed as an "average" of a group of objects, and at the same time the group itself is being formed. One might well ask whether, given a BP, Phaeaco discovers on its own the two groups of six boxes by following its general group-formation principles. The answer is "no", because Phaeaco is designed to know what a BP is, and how to handle it. Thus, the identification of the notion "group of six boxes of a BP side" is not actively pursued by the program. What is actively pursued is a visual pattern that summarizes the six boxes on either of the two sides. In some cases, as will be explained later, more than one visual pattern can be formed on each side.

Three distinct mechanisms that appear to be involved when a person attempts to solve a BP[82] are discussed in the following subsections.

[82] Personal observation, in agreement with the intuitions of other, experienced BP solvers. Special devices would be required to confirm this observation under controlled conditions, such as an eye-tracking apparatus.

11.1.1 First mechanism: hardwired responses

There are exactly two BP's in Bongard's collection that can be solved by means of mechanisms hardwired in the human brain. These are BP #3 ("outlined vs. filled", Figure 1.2), and BP #1 ("nothing vs. something", Figure 1.5).

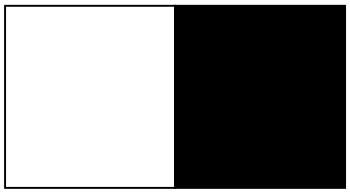


Figure 11.1: Contrast between colors, white vs. black

Figure 11.1 illustrates the principle behind the solution of such problems. The human visual cortex includes the color-specializing area V4 (e.g., Zeki, 1993), which is stimulated immediately and involuntarily given two contrasting colors. Although the response in V4 is instantaneous, a few seconds are required before subjects reach the answer for BP #3 (average response time 7.9 s; see Appendix A). Possibly subjects are slowed down by multiple high-level perceptual processes that occur simultaneously, such as the perception of the frames of the visual boxes that interfere with the contrasting colors, and a large number of other features (shape, size, position, etc.) that compete for attention. In principle, verification of the solution (i.e., examining the boxes one by one) is unnecessary for such BP's; but in BP #3 it is possible that subjects perform a small amount of verification, because half of the boxes contain very small objects that do not contribute significantly to the perceived color. In BP #1, the solution is perhaps initially hinted at by the contrast between the whiteness of the left side and the existence of black regions and black lines on the right side, but it is expressed as

"nothing vs. something" (or "no object vs. some object", as Phaeaco puts it), because it is not logical to see this problem as a contrast between white and black.

There are other BP's, beyond Bongard's collection, which also benefit from the hardware of the brain. Examples are: BP #157 (reversing the foreground and background colors), BP #158 ("some slope vs. a different slope"), and BP #196 ("light-colored texture vs. dark-colored texture"); see these BP's in Appendix A. A hypothetical BP that includes animated figures on the left and still figures on the right would be another such example, benefiting from area V5 of the visual cortex, which detects motion.

Phaeaco, like human solvers, reaches the solution of BP#3 very quickly, and usually (98% of the time) prints the answer without verifying it for each of the 12 boxes. But the mechanisms that Phaeaco uses to achieve this apparent human-like performance differ from the above-mentioned mechanisms of the human brain.

Specifically, as was mentioned in the introduction to this chapter, Phaeaco constructs a pattern that summarizes each side. The variety of shapes on each side of BP #3 results in the formation of a single pattern per side. To understand why this is so, consider the abstractions in Figure 11.2.

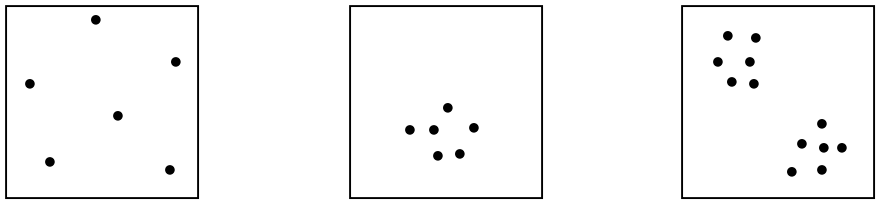


Figure 11.2: Different cases of group formation depending on the distances between points

Each of the points in Figure 11.2 stands for the representation of the contents of a BP box. The distance between two points corresponds to the psychological

difference between the contents of two boxes. In Figure 11.2a the points are quite distant from each other, but they form a single group; therefore they can be summarized by a single pattern-point, which can be imagined at the barycenter of the group. This situation is analogous to the right side of BP #1, which includes boxes with shapes that appear as different as shapes can be. In Figure 11.2b the points do not differ as much, but they still form a single group. This is similar to either of the two sides of BP #3. Finally, in Figure 11.2c the points form two distinct groups. There are very few BP's with boxes on one side that can form more than one group. An example of such a BP is shown in Figure 11.3.

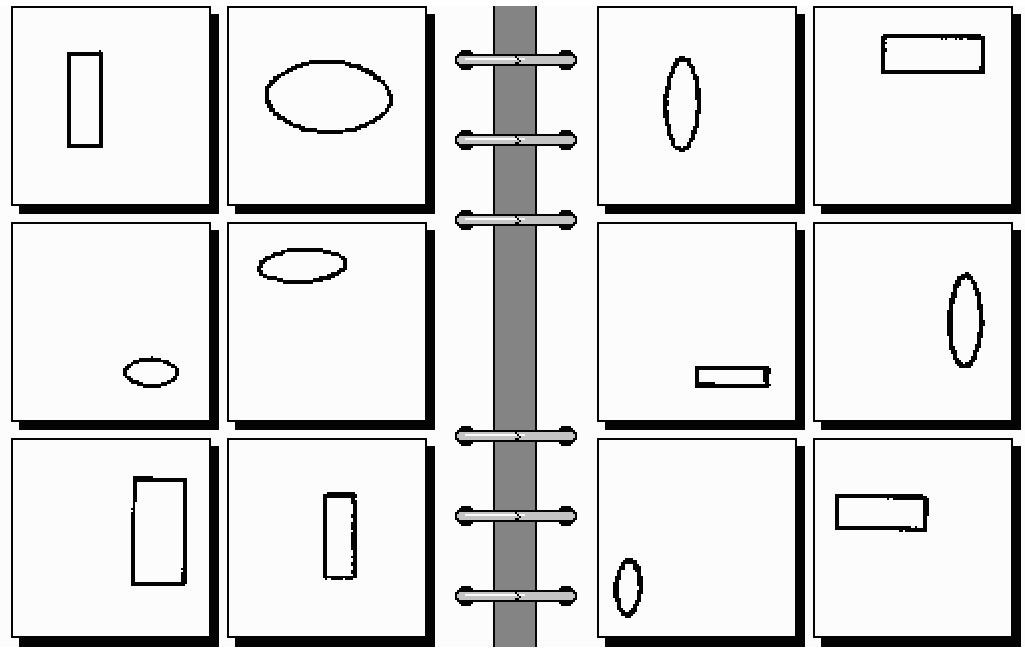


Figure 11.3: BP #13, in which two groups (patterns) of shapes are perceived per side

On the left side of BP #13 in Figure 11.3 two distinct patterns can be discerned: vertical rectangles and horizontal ellipses. Similarly, on the right side there are vertical ellipses and horizontal rectangles.

Consider again the case of BP #3. After Phaeaco takes a cursory, overall look at all 12 boxes in parallel, it constructs a single pattern for each side. The two patterns typically have the structure shown in Figure 11.4.

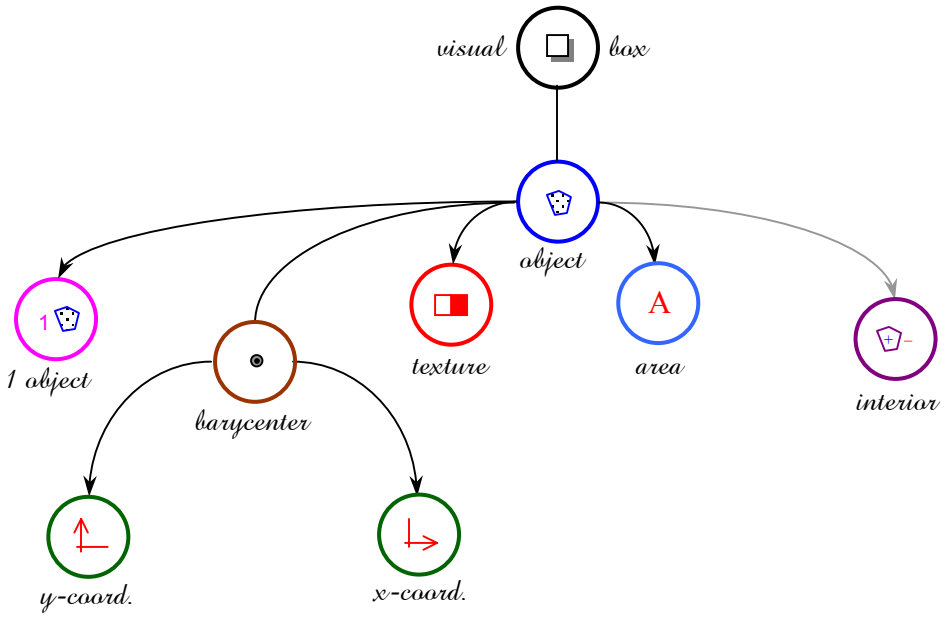


Figure 11.4: Pattern of left and right side of BP #3

Figure 11.4 shows only the nodes included in the structure of the patterns, but the values of the featural nodes generally differ between the two patterns. For example, the average area of the left-side objects is different from the average area of the right-side objects. However, the statistical comparison of features involves also their variances, and because the areas of the two patterns have large variances, they do not differ significantly. But the comparison of textures yields a different result: the sample of the left side consists of six identical values (all outlined, zero variance), and the same is true for the sample of the right side (all filled, also zero variance). The fact that the variance is zero in both samples

causes Phaeaco to halt immediately and output the correct answer without verifying it on any of the 12 boxes.

On first thought, it might appear inappropriate for Phaeaco to employ a mechanism ("zero variance") very different from the one humans use, only to solve BP's in a time comparable to human performance. But this is yet another example of Phaeaco being faithful to its philosophy, summarized in Figure 4.9: humans and machines differ fundamentally at lower levels; the challenge is to make machines converge with humans at higher levels.

Phaeaco homes in quickly on differences in textures for an additional reason: codelets responsible for registering the texture of objects have high urgencies, and so textures are among the first features that are seen. This bias was built into the architecture with the knowledge that perceiving and contrasting colors is similarly hardwired in humans.

The previous discussion on comparing samples with zero variance suggests a way that, in theory, should cause Phaeaco to respond in an unhumanlike way. For instance, consider a BP in which the same object (down to the last pixel) is repeated six times on the left side, and a *slightly* different object is repeated six times on the right side. Conceivably, the sample of object areas on either side should have a variance equal to zero, and because the average values of the two compared areas would be different (however slightly), Phaeaco would halt immediately and announce with absolute certainty the difference in sizes, whereas the human eye would have trouble discerning the minute difference.

In practice, however, this does not happen. Recall that Phaeaco's processing is nondeterministic, because it is randomized at the earliest possible stage (§10.3.1). Thus, although the areas of the six objects on each side would be very similar, the probability that they resulted in an identical number would be practically zero.

11.1.2 Second mechanism: the holistic view

Hardwired responses aside, experienced solvers seem to employ an initial strategy when confronted with a BP, in which they conduct a panoramic overview of the problem for a brief period of time (perhaps 2–3 seconds), without focusing for too long on any box in particular, in an attempt to see differences "jumping out" at them, as quite a few people report. If no difference becomes apparent in this brief "holistic" comparison of the two sides, they then resort to an "analytic" examination of individual boxes, which is described in the next subsection.

Phaeaco acts similarly. It processes all 12 boxes in parallel, distributing its time among the boxes in such a fine-grained manner that for all practical purposes the processing of the different boxes appears to occur simultaneously. During this time, patterns for the two sides are formed[83] and compared against each other, applying the group formation methods of chapter 8. If more than one pattern is generated per side (as in BP #13, Figure 11.3), their disjunction is represented by placing the patterns under a Necker view node (§7.4.11).

The comparison might reveal that the patterns differ in structure, as in BP #6 ("triangle vs. quadrilateral", Figure 1.1), or BP #97 ("triangle vs. circle", Figure 2.7). Another possibility is that something exists in one pattern, but is missing from the other pattern, as in BP #1 ("nothing vs. something", Figure 1.5) and BP #5 ("no curve vs. some curve", Figure 5.6). A third, and more common, scenario is that the value of a feature on the left side differs from the value of the same feature on the right side, as in BP #2 ("large vs. small", Figure 1.3). The case of differences in a feature value is worthy of further consideration.

---

[83] A subtle technical point here is that the representation of each box is continually updated as long as retinal processing takes place, which raises the question of the proper time to consider all six representations of the boxes and form one or more patterns out of them. Phaeaco waits until the activation of a box (as explained in §7.2) drops below a threshold. The threshold is low enough to allow a more-or-less complete representation of the box contents to be formed.

Consider again BP #2 (Figure 1.3). Table 11.1 lists the areas in pixels of each of the 12 objects, as they were found in one particular run by Phaeaco. (These values differ from one run to the next, since the processing is nondeterministic, but they have roughly the values given in the table.)

Box	Area	Box	Area
1A	1348	2A	128
1B	1662	2B	134
1C	1986	2C	244
1D	1994	2D	104
1E	2491	2E	133
1F	2570	2F	164
Average:	2008.50	Average:	151.17
Std. dev.:	470.13	Std. dev.:	49.33

Table 11.1: Boxes and areas of objects of BP #2

The two samples can be compared using the methods and formulas of §8.2. They can also be approximated by two Gaussians, as shown in Figure 11.5.

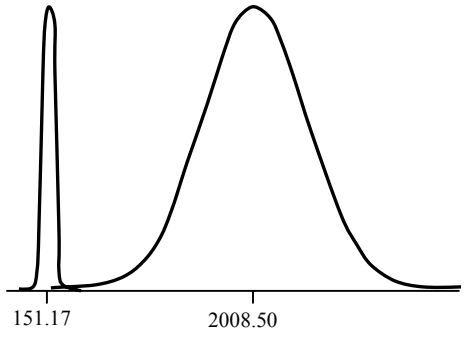


Figure 11.5: The two distributions of areas shown as two Gaussian-like curves

The curves are plotted in Figure 11.5 only for illustration purposes. In reality the samples are not guaranteed to originate from normal distributions, because the areas of objects can be arbitrary. But Student's *t* distribution (which is used by Phaeaco, see §8.2.1) generally resembles a normal distribution. Also, the two curves are shown with equal altitude in Figure 11.5, which is clearly wrong. Since the area under each curve must equal 1, the second curve must be very flat; but if it were plotted realistically, it would not be perceived as the Gaussian-like curve that it is.

In spite of these liberties in Figure 11.5, the salient point is the minute area of overlap of the two curves. The distributions of the two areas (as suggested by the two samples), are easily separable, because their intersection is very small. When this happens, it is easy for Phaeaco to direct its attention to the idea "difference in areas". In contrast, if the two suggested distributions were as depicted in Figure 11.6,[84] then the probability that Phaeaco would notice the difference is much smaller.

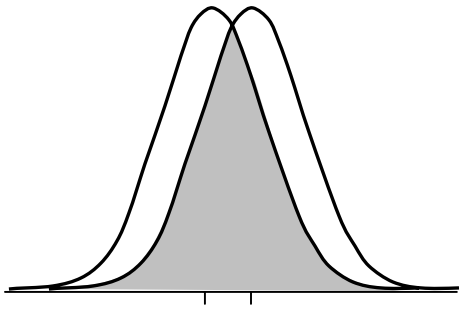


Figure 11.6: Two difficult to separate distributions

How can Phaeaco's attention be directed to one feature or another? The answer is that there are *codelets* that visit the structures of the patterns, examining

[84] An example would be the distributions of the coordinates of the centers of the objects in BP #2.

their features. These codelets (and their Coderack), which are distinct from those that create the initial representations in each box (as described in chapter 7), belong to a higher conceptual level, specifically concerned with the solution of BP's. In other words, these are Bongard-specific codelets. Some codelets are assigned the task of identifying features[85] and computing the probability that their distributions are distinct, as described above. Other codelets follow up the initial ones, but with an urgency proportional to the probability of a difference in distributions, as already computed by the initial codelets. Thus, a "sharp" difference has a higher probability of being noticed than a "fuzzy" one, a calculation that considers both the inherent significance and the current activation of the Platonic feature in LTM (Hofstadter, 1995a, p. 226).

Once a featural difference is noticed, Phaeaco forms the idea that this might be a solution of the given BP. To "form an idea" for a solution in the BP domain means to construct a pair of representations that describe the idea. For example, if the idea is that the sizes of objects differ, as in BP #2, then a pair of nodes of the areas of the patterns suffices. Figure 11.7 shows the representation of this very simple idea for a solution.

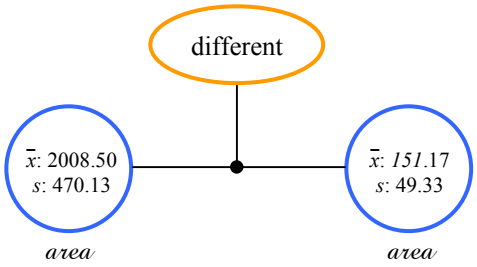


Figure 11.7: An idea for a solution in BP #2

[85] The urgency of such codelets depends on the salience of the corresponding Platonic features in LTM. For example, the urgency of a codelet that compares textures on the two sides generally is higher than the urgency of one that compares the *x*- or *y*-coordinates of barycenters.

Ideas for solutions can be far more complex. For example, a solution that contrasts two different patterns (e.g., "triangle vs. circle", BP #97) has the entire patterns connected in the same way the two nodes are connected in Figure 11.7. A solution based on the presence of something on one side and its absence on the other side has the structure that is present linked to a special "missing" node.

Next, Phaeaco attempts to verify the validity of the idea by examining each of the 12 boxes in sequence. During the verification of a featural difference, the following condition is examined: it is not sufficient that the features differ greatly, because in BP solutions it must also be true that the ranges of the two samples do not overlap. For example, following the samples of Table 11.1, the range of the large areas is from 1348 to 2570, and the range of the small areas is from 104 to 244. Thus, the minimum value of the higher range (1348) is larger than the maximum value of the lower range (244). If this were not the case, the idea would be rejected. This observation provides a justification for explicitly storing the minimum and maximum values in the structure of a statistical table (Table 7.1).

During the verification of the solution, Phaeaco takes a final look at each box that is being examined. Thus, if the solution is found at the end of the holistic stage, each box is seen twice: once during the holistic stage (in parallel with all other boxes), and once more during the verification. The resulting two representations are matched, and a single pattern is made, akin to a sample of two elements. Phaeaco does this in order to improve the representation that it obtained from the single observation during the holistic look of the problem. Recall that Phaeaco's vision is not as sharp and accurate as is human vision, so it is advantageous for it to examine each box once more before issuing a judgment.

If the idea fails to be verified on the pattern thus obtained, it is rejected, and Phaeaco enters the next, "analytic" stage of trying to solve the BP.

11.1.3 Third mechanism: the analytic view

If the holistic stage fails to produce a solution (either because the verification of an idea failed, or because the patterns of the two sides were identical, and so no idea for a solution was generated), Phaeaco enters its analytic stage. Individual boxes are selected nearly randomly (more on this later), and reexamined in an effort to come up with fresh ideas. But how does Phaeaco come up with new ideas when the input is given and unchanging?

Part of the answer stems, once more, from Phaeaco's nondeterministic nature. When a box is watched multiple times, its contents can be seen in different ways. These differences can be consequences of the processing that took place in the other boxes, which can prime concepts in LTM and thus influence the way the contents of the current box are interpreted. In addition, as was explained in chapters 7 and 10, a single look at a box does not result in a complete representation including everything there is to be perceived in the box. On the contrary, every iteration reveals only a partial view of a box's contents, but each partial view enhances the pattern created by the cumulative effect of all the looks.

Suppose, however, that the representations of the boxes do not change at all, no matter how many times they are seen and re-seen. Even then, Phaeaco would still try to come up with new ideas for a while. (Later it will be explained what makes Phaeaco give up.) Codelets are created at this stage to look at representations in boxes and come up with ideas. For example, if there is a triangle in a box, a codelet might notice the numerosity node representing "3 line segments", and create the idea: "Could this problem be about counting lines?" If this idea proves unfruitful, probable future ideas include: "Could this problem be about counting things in general?" and "Could this problem be about 3-ness?"

The urgency of such codelets causes specific ideas to be tested first and their generalizations later, after the specific ones have failed to solve the problem.

A problem that is also addressed by Phaeaco is the avoidance of verifying ideas that have already been verified and shown to be wrong. It is natural for an intelligent system to make errors, but it is an indication of mindlessness to repeat errors uncritically (Hofstadter, 1985, pp. 526-546). It is not wrong or unhuman to regenerate an idea — human solvers seem to keep on coming back to the same failed ideas all the time while solving BP's — but it is wrong to seriously consider and proceed to verify a failed idea. Phaeaco solves this problem by linking all verified-and-failed ideas to a master node (Figure 11.8).

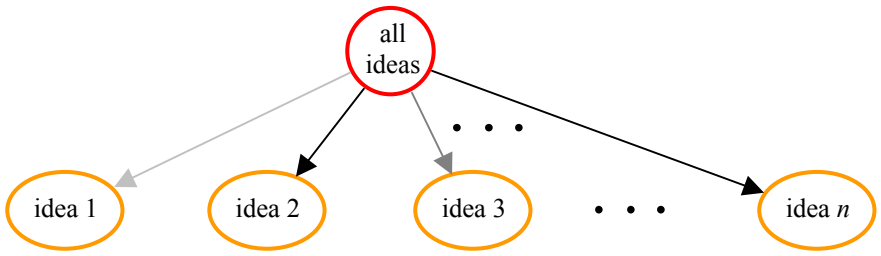


Figure 11.8: Representation of tested ideas

Some of the links connecting the master node ("all ideas") to individual ideas in Figure 11.8 are shown grayed, reflecting the lower strength of these links. This is important, because what is shown in Figure 11.8 is not a traditional linked list, but a more elaborate structure that suggests the solution to an interesting cognitive problem. To illustrate this, consider the following example.

A person is asked to say the name of a month of the year, at random. The person answers: "October". The question is repeated with the condition that the new random month must be different from the previous one, and the person answers: "April". After some repetitions, when the number of randomly generated

months approaches (but is not yet equal to) 12, the person will probably pause briefly, trying to figure out which month names have not yet been used. During this period of hesitation the person will probably recall a few already generated names, and will subliminally reject them with a thought that, if expressed in words, would amount to: "No, I have already said that." This is the problem that the representation in Figure 11.8 helps to solve. The mechanism works as follows.

When an idea is generated by a codelet it is compared in parallel[86] with all the ideas linked to the master node, as in Figure 11.8. Although the general pattern-matching procedure (§8.2) is used for the comparison, the idea is rejected only if it is identical to one of the verified-and-failed ones. If an identical idea is found, the strength of the link of that idea is enhanced by an "injection" according to the way activations increase (§7.2), and the idea is not added to the list. Otherwise, the idea is added to the list, connected to the master node with a link of strength 1 (the maximum value), and the activation of the master node is enhanced by a small amount (a discrete step of enhancement, §7.2).

The storing of strengths on the links implies that some of the generated ideas can be forgotten, because their link strengths dissipate as time goes by and can become effectively zero. In practice, solution ideas cannot be forgotten in the short period during which a BP is solved, but the mechanism is general.

Also, suppose the person performing the above-mentioned month-listing experiment, after succeeding in uttering all 12 months, is given the following request: "Now forget that you just listed all the months, and let us repeat the experiment from the beginning: please produce a new random list of the months." The person is in a position to succeed in this new task, feeling very small interference from the previous attempt at producing a list. This ability is also

[86] But parallelism is of course implemented by codelet interleaving on sequential machines.

important in Phaeaco's case: when a new BP is presented, the master node of ideas of the previous BP is eliminated in the Workspace, along with its linkages, and a new master node is created for the new problem. The small interference a person would feel in generating a new list can be explained by the priming of ideas in the LTM. However, Phaeaco at present does not implement the notion of copying ideas to the LTM after a BP is solved. (It can prime concepts in LTM, but does not store ideas, i.e., solutions of BP's.)

A final question is: when does Phaeaco stop coming up with new ideas and quit trying to solve the BP? The answer is, when it "feels bored". Phaeaco's boredom in solving BP's is implemented in a very simple manner: it is the activation of the master node. Recall that each new idea enhances the activation of that node by a small amount, but the lack of new ideas causes the activation to drop gradually. If the activation drops below a certain threshold, Phaeaco gives up.[87] By adjusting the value of the threshold, one can make Phaeaco appear more or less insistent in solving BP's, just as human solvers differ in this aspect according to their personality traits.

In the beginning of this subsection it was mentioned that in the analytic stage boxes are selected and examined again "nearly randomly". The selection is not completely unbiased. Boxes with simpler contents are selected more often. This is because experienced BP solvers report feeling that it is easier to find the essence of a BP in a box with simple contents than in a complex one. The simplicity of the contents of a BP is reflected directly in the simplicity of its representation: an elementary graph-visiting process can estimate the simplicity of a representation (e.g., the number of nodes can serve as a crude estimate).

---

[87] Therefore, instead of "boredom", the opposite term "motivation" appears more appropriate: when the activation drops below a certain threshold, Phaeaco feels no motivation to continue.

Finally, the previous brief description of the process of verification might erroneously suggest that once an idea fails to be verified on a single box, it is immediately discarded. But what if 11 of the 12 boxes strongly suggest one idea, but the twelfth box fails to confirm it? Consider Figure 11.9.

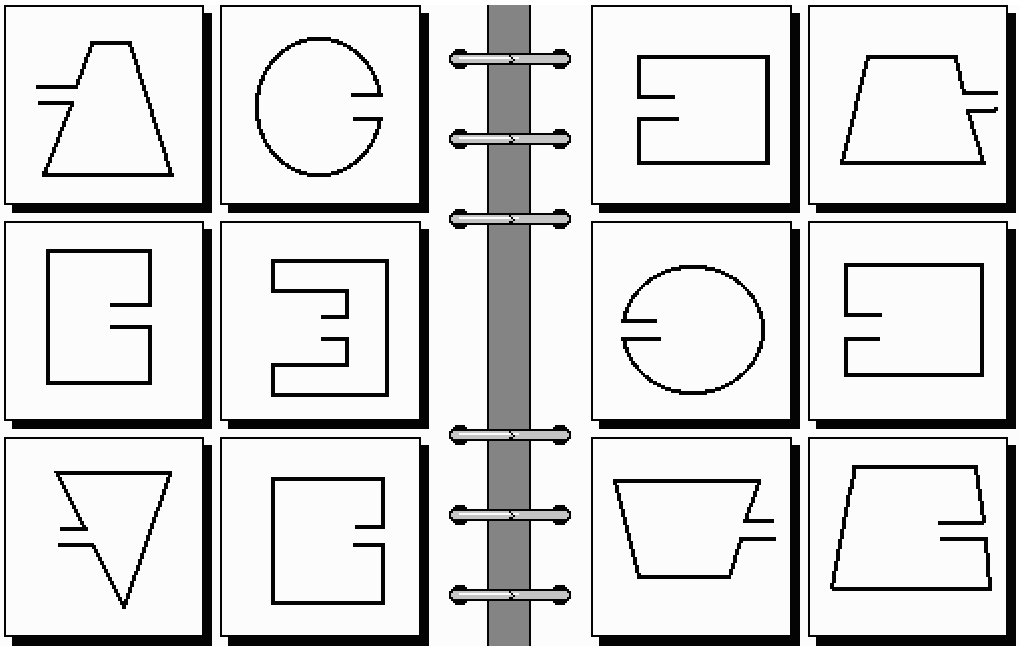


Figure 11.9: The "trickster" BP #192

Most solvers perceive the direction of the orifice in each of the objects in BP #192 (Figure 11.9). On the left side, it looks like the orifice points to the left, whereas on the right side it points to the right. Or does it? A closer look at the bottom-right object of this problem reveals that its orifice points to the left. Could this be an error? Most solvers focus their attention again and again on this box, unwilling to discard without further thought an idea that seems to work so well on all the other boxes. Some conclude that the designer made an error. Others accept the ineffectiveness of this idea, often proceeding to find the real solution, which is

that the objects on the left are elongated in the vertical direction, whereas the objects on the right are elongated in the horizontal direction.

Phaeaco does "the same". If most of the boxes (even on a single side, e.g., four or five out of six) have confirmed one idea, but the remaining boxes (one or two) do not seem to confirm it on the first verification effort, Phaeaco will persist in trying to verify the idea on the remaining boxes for a while, looking at them again and again in an attempt to find by chance an alternative view (representation) of their contents. The stronger the confirmation of the idea from the remaining boxes, the more Phaeaco will attempt to see the failing box or boxes. Also, recall that Phaeaco's low-level vision is not as sharp as the vision of the human eye, so it is entirely possible that some hard-to-see feature was missed in the earlier attempts on a box. By looking repeatedly at the input, Phaeaco increases the probability of noticing what it missed.

11.2 What Phaeaco can't do

There are a large number of perceptual primitives as well as abstract principles of visual cognition that must be implemented in any system that aspires to solve all 100 original BP's (and, consequently, most of the BP's in the extended collection of 200, listed in Appendix A). The most important of these primitives and principles are discussed in the present section. But it must be emphasized that these have been omitted only from the present implementation of Phaeaco; their omissions do not constitute basic architectural flaws that cannot be addressed in

future enhancements.[88] The present section should be construed precisely as a list of suggested implementational enhancements.

11.2.1 Conjunction of percepts

Often it is necessary to combine two or more notions in order to arrive at the solution of a BP. A typical example is given in Figure 11.10.

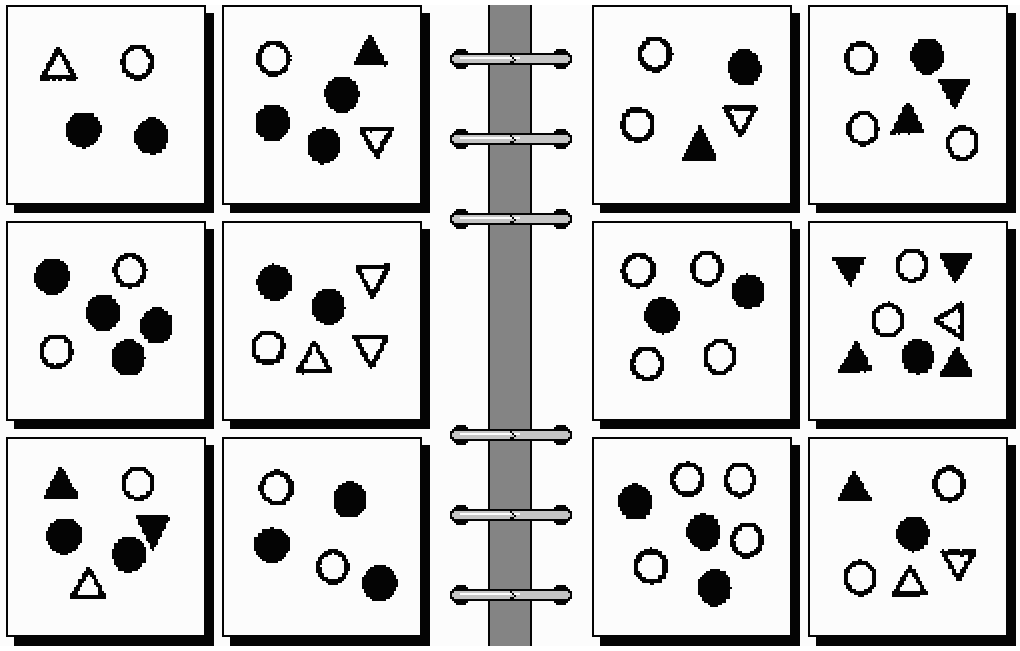


Figure 11.10: BP #28, requiring a combination of simple percepts

The solution of BP #28 (Figure 11.10) is that on the left there are more filled circles than outlined circles, whereas the relation is reversed on the right side. Each box of BP #28 uses several concepts ("outlined", "filled", "triangle", "circle", etc.), and the solver must select a particular (arbitrary) combination of

[88] The title of this section is an allusion to Hubert Dreyfus's "What Computers [Still] Can't Do" (Dreyfus, 1972; 1992), an early philosophical criticism of A.I. But, unlike Dreyfus's work, which conveyed an implicit message of what computers will *never* do, the present section aims merely to suggest tasks that future BP-solvers can (and must) do.

them to arrive at the solution. The arbitrariness and the combinatorial nature of the task make such BP's rather unattractive, and present an understandable obstacle to human solvers. (Not surprisingly, none of the 21 subjects that attempted to solve this problem found the solution.) Nonetheless, being able to combine percepts, consider the resulting groups of objects, and perform a rudimentary mental combinatorial search is an ability the average human solver has.[89] This ability is also related to the notion of "noise", considered next.

11.2.2 Screening out "noise"

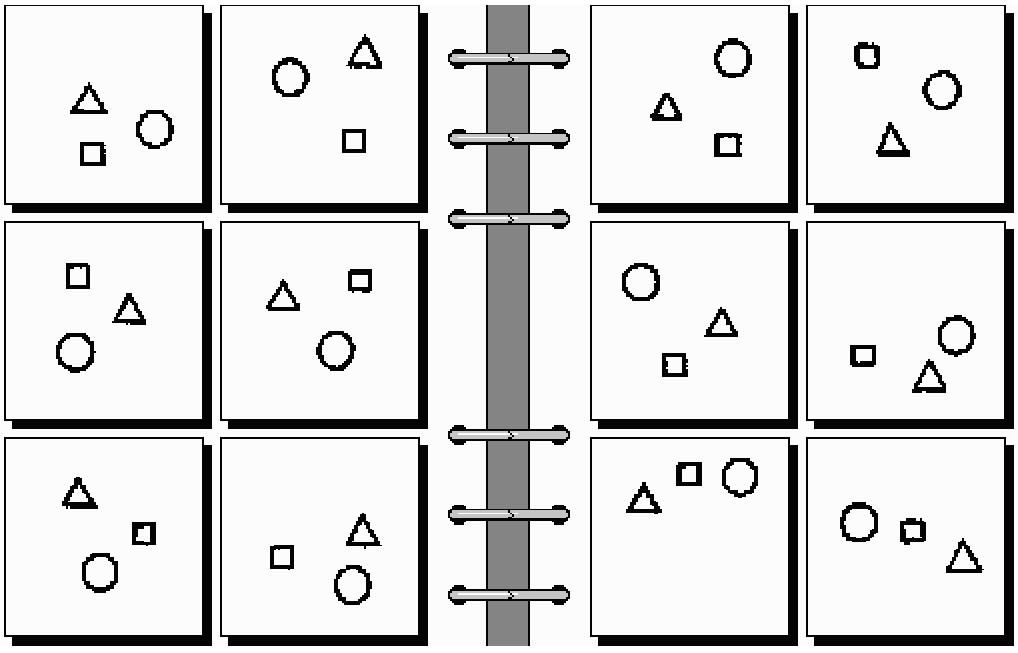


Figure 11.11: BP #37, where the squares are mere distractors

Almost every time a problem such as BP #28 is presented, a related cognitive ability is also employed: concentrating only on a part of the available input, and

[89] BP #28 should not be used as evidence against this statement, because there are easier BP's that support it, including #26, #32, and #81.

treating the rest as "noise". Naturally, noise appears essentially in every BP, because it is virtually impossible to construct a BP in which *every* available percept participates it its solution. Sometimes, however, noise is purposefully inserted in the input by the designer, turning an otherwise rather easy solution into a harder (and sometimes more interesting) one.

Figure 11.11 shows BP #37, in which the squares are superfluous. Its solution is: "triangle above circle vs. circle above triangle" (where "above" means that the *y*-coordinates of the centers of the objects are compared). This problem appears in Bongard's collection immediately after one that has exactly the same solution but lacks distractors. When people are presented the two problems in Bongard's order (BP #36, BP #37) they usually solve easily BP #37, having been primed with the solution of BP #36. But if the two problems are not presented in sequence, BP #37 can prove to be quite hard to solve. In the experiment reported in Appendix A, subjects were shown BP #37 only after 38 more BP's had been presented following BP #36. As expected, although BP #36 proved relatively easy to solve (correct: 23; incorrect: 2; no answer: 5), BP #37 turned out to be quite difficult (correct: 3; incorrect: 5; no answer: 17). In general, adding noise can turn some otherwise easy BP's into extremely difficult ones.

11.2.3 Applying a suspected solution on all boxes uniformly

Sometimes a solution is suspected on some (perhaps most) boxes, but the contents of a few other boxes do not lead naturally to this solution. In other words, if the objects in these exceptional boxes were seen in isolation, it would not be easy to think of the statement that solves the BP, because it is only in the context of the entire problem that a different perception of these objects is evoked. An example of this principle is shown in Figure 11.12.

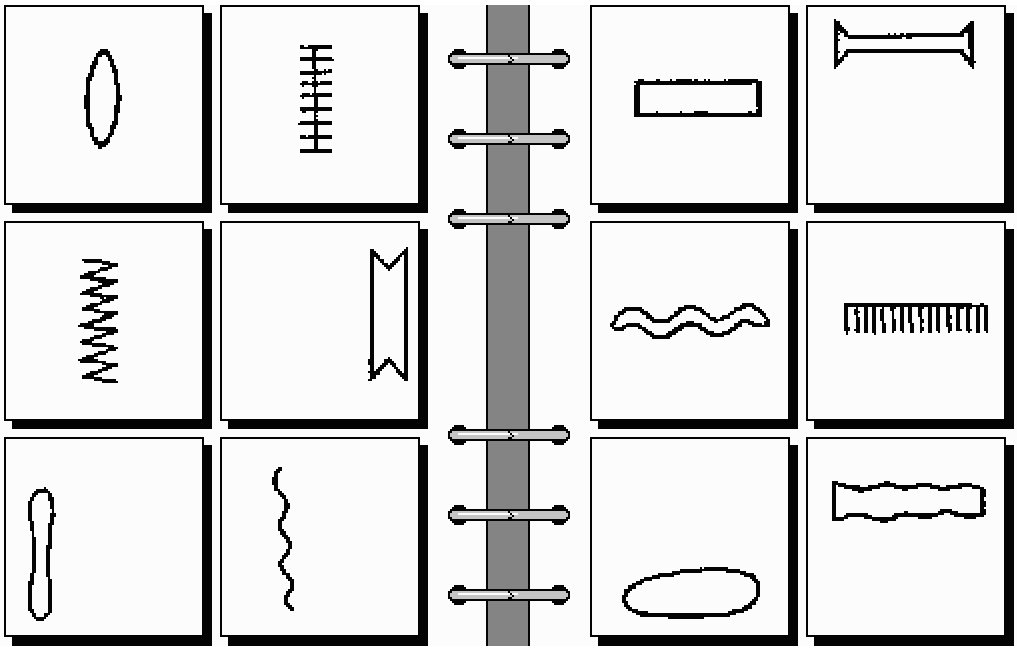


Figure 11.12: BP #7, necessitating re-parsing of the contents of some boxes

Some of the boxes in BP #7 (Figure 11.12) contain elementary objects that, out of context, could be described very simply as "an ellipse", "a rectangle", and so on. But other boxes contain complex objects, some of them made out of a large number of lines, which leads Phaeaco to apply the shrinking algorithm (§10.3.16) and to derive their convex hulls. These actions generally suffice to allow Phaeaco see the complex objects as "something elongated", and even to perceive their direction, which is vertical on the left, and horizontal on the right. Thus, in order to reach the solution, the solver constructs a procedure that is applicable in many boxes:

- Shrink the object and/or derive its convex hull
- Observe that the result is elongated
- Perceive the direction of the primary axis of the elongated object

It is necessary to be able to apply uniformly, to all boxes, a procedure that was created "on the fly". In this way, the directions of simple objects, such as the ellipses and rectangles in BP #7, can also be perceived.

A different example will illustrate the need to have the ability to "complexify" the descriptions of boxes when simpler descriptions are perceived at first.

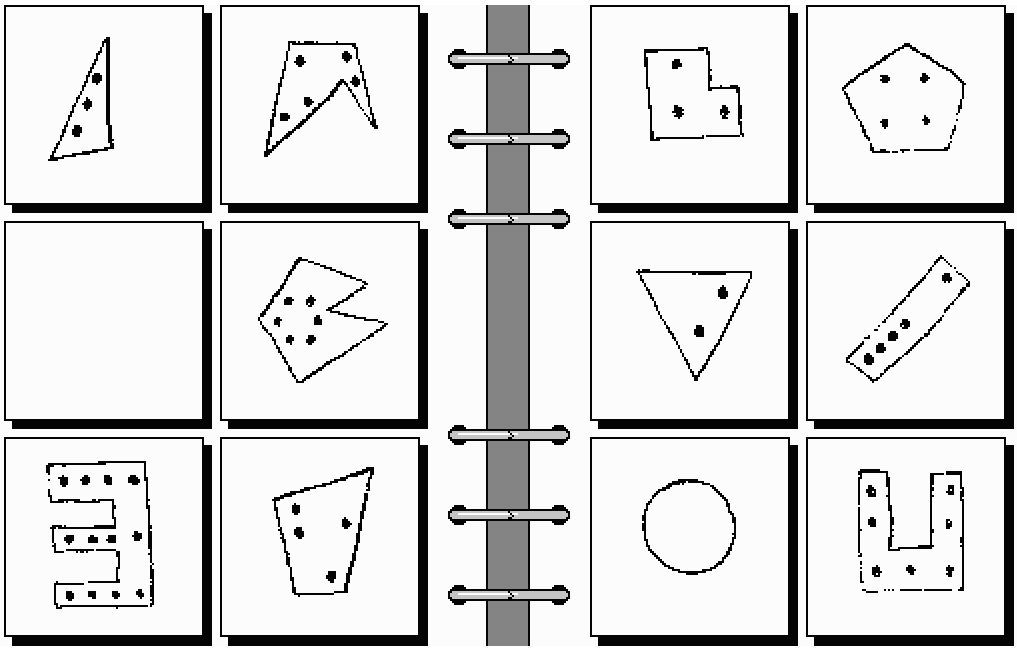


Figure 11.13: BP #137, where there is something in nothing

Consider BP #137 (Figure 11.13). The empty box on the left side is initially perceived simply as an empty box. But after one realizes that the solution involves comparing the number of dots with the number of lines that make up the closed figure, one can easily see that the empty box stands for the relation: 0 = 0.

Such "forced descriptions" appear all the time in BP's, especially in those that require the solver to re-interpret a single object as a degenerate group made of a single object (see BP's: #81, #89, #90, #156, #166, and #167).

11.2.4 *Pluralitas non est ponenda sine necessitate*

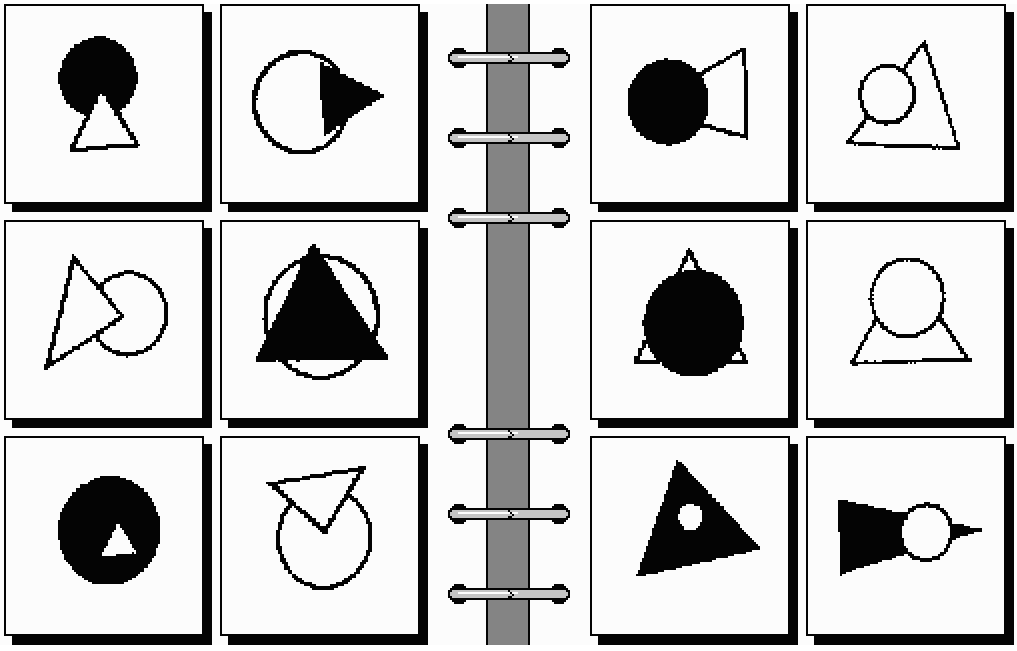


Figure 11.14: BP #46, in which Ockham's razor cannot be ignored

Consider BP #46 (Figure 11.14). Human solvers effortlessly do something that any programmed solver, such as Phaeaco, must be instructed about explicitly and painstakingly: they see *a circle* hidden under a triangle on the left, and *a triangle* hidden under a circle on the right. But "in reality" there are no such shapes. Instead of circles on the left, the raw data contain arcs and circular sectors (plus a filled circle with a triangular hole). One might claim that in boxes I-A, I-C, and I-F there is not even a triangle: instead, there is a nearly 270°-wide circular sector and a line-string, two of whose three line segments are collinear with the two radii of the circular sector (Figure 11.15a). Why does the human eye never come up with a description like this, or any of the other ones shown in Figure 11.15, as a first impression?

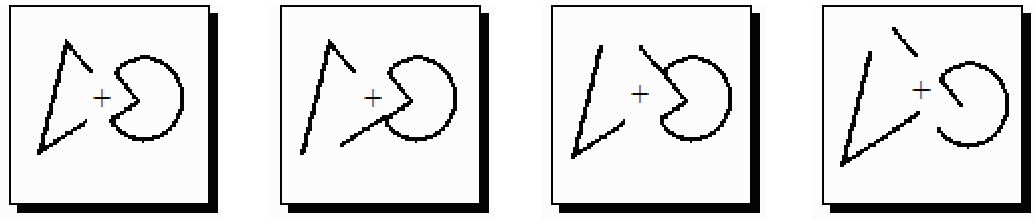


Figure 11.15: Unnatural parsing possibilities for box 1C of BP #46

The answer "because the description 'triangle + circle' is the only one that makes sense" begs the question: why does it make sense? A more appropriate answer is that the description "triangle + circle" is of minimum length. To state it otherwise, Ockham's razor can eliminate all possibilities shown in Figure 11.15 because "triangle + circle" is the most parsimonious description of all.[90]

To arrive at the "minimum length description" a viewer must not only have visual patterns stored as concepts in LTM, but also be able to perform visual operations such as continuing curves along their curvature (beyond the points where they are visibly interrupted), and extending straight lines up to the point of their intersection. Such operations should appear "irresistible" if they result in shapes that match very well with LTM concepts. This ability, by the way, also suggests a way to explain the well-known Kanizsa illusion (Figure 11.16).

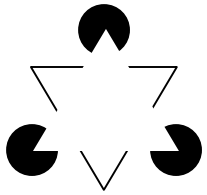


Figure 11.16: The Kanizsa triangle illusion

[90] It is parsimonious assuming that the viewer is familiar with the visual patterns of a triangle and a circle; if the viewer were from another world, where "pacman-like" and "angular-C-like" objects were more familiar than triangles and circles, perhaps the parsing of Figure 11.15a would be considered more parsimonious.

11.2.5 Meta-descriptions

Related to the minimum-length description ability (or Ockham's razor) is the ability to perceive descriptions (representations) themselves and infer their properties at a meta-level (Hofstadter, 1979, pp. 656-661). For example, the decision regarding which of two descriptions is shorter implies "looking at" and measuring representations. But this is not unusual; indeed, there are many other situations that would call for representations to be examined. Another example is the perception of the recursive "depth" of a relation. Besides BP's #70 and #71 (Figures 1.12 and 1.13, respectively), BP #186 in Figure 11.17 also demonstrates a relation ("is made of") generating multiple levels of description.

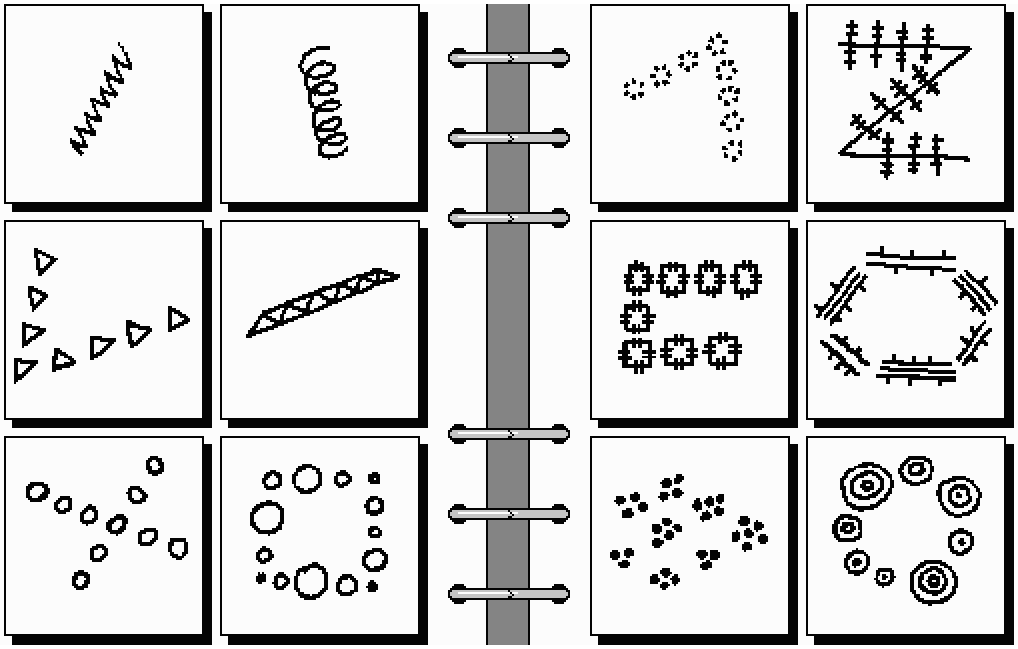


Figure 11.17: BP #186: one level of detail vs. two levels of detail

Being able to reflect upon one's own thoughts is a hallmark of human-like intelligence (see also §9.2). Metacat (Marshall, 1999) and Letter Spirit (Rehling, 2001) are systems that reportedly have this ability.

11.2.6 Figure–ground distinction

The problem of distinguishing contents from background (or making a figure–ground distinction, as it is usually called) was mentioned in the context of retinal processing (§10.1.1). But this problem is primarily conceptual, which is why the heuristic described in §10.1.1 appears inadequate. The only BP that addresses this problem at a conceptual level is BP #98 (Figure 11.18).

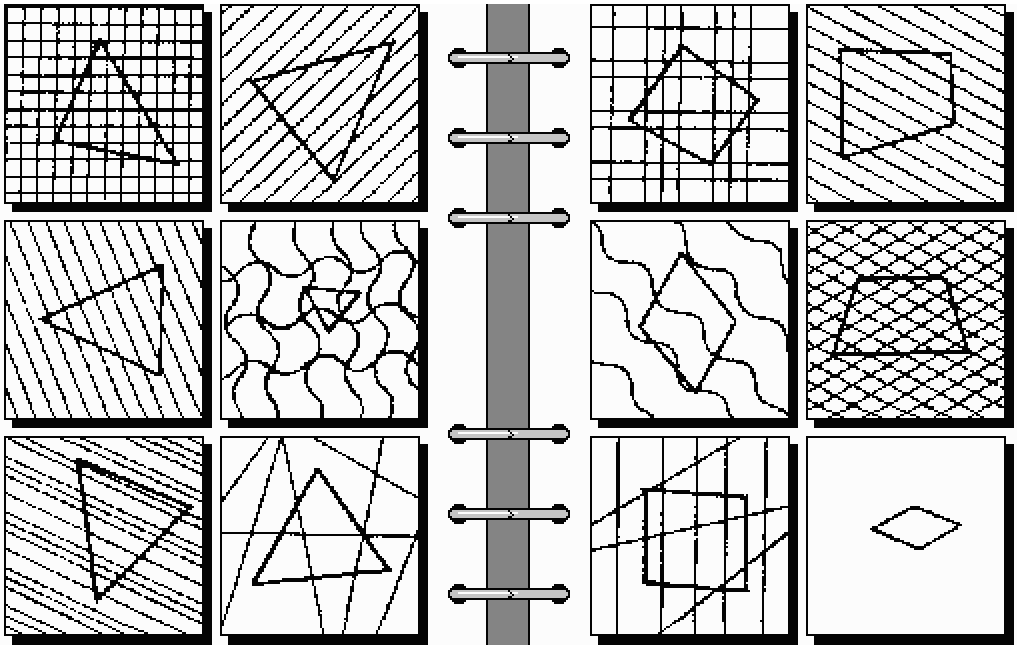


Figure 11.18: BP #98, an exercise in figure–ground distinction

BP #98 would be a repetition of BP #6 ("triangles vs. quadrilaterals")[91] if it were not for the patterned background that interferes with the "main" shapes.

Essentially, Phaeaco already possesses the most important mechanisms for solving this problem. The key observation is that of all the lines (both straight and curved) in a box, some "belong together" because they have many common properties, and so they form the background. The remaining lines do not belong to the previous larger category and thus make a category of their own (the foreground). It is a categorization problem. Unfortunately, there are many more details that prevent Phaeaco's current implementation from reaching the solution in this problem,[92] but the main principle is already in place.

Interestingly, this type of perceptual problem has been used in recent years to prevent machines from creating internet accounts, participating in discussion forums (thus flooding discussions with advertisements), etc. Typically, the user is presented with input as shown in Figure 11.19. If the user cannot type the alphanumeric characters in a provided data entry field, it is deemed a computer.

Figure 11.19: Typical input expected to baffle computers

[91] BP #6 was answered correctly in an average of 19 sec (26 subjects; wrong: 2; no answer: 3). But, surprisingly, BP #98 took only 12 sec for those who solved it (9 subjects; wrong: 3; no answer: 4). This might be because BP #6 appeared early (3rd), whereas BP #98 appeared close to the end (98th), so only the better solvers reached it, who meanwhile had gained experience in the BP-solving task.

[92] For example, Phaeaco should not get bogged down in the countless intersections of lines, but should be able to see the similarity in wavy lines (currently it cannot), and should insist on applying the correct idea discovered in some boxes to the most difficult of the other boxes (I-D, I-F, and II-E), as discussed in §11.2.3.

Variations of this type of problem exist, but they are all based on the idea of figure–ground distinction. However, a "Phaeaco" that was free from the burden of having to solve a BP and was coupled with an optical character-recognition module could easily solve this problem. The prediction is that web-page designers in the future will have to resort to cognitive problems that exploit more quintessentially human abilities if they want a test that will automatically distinguish humans from computers.

11.3 Summary

The three mechanisms for solving BP's that are assumed to be employed by human solvers (hardwired, holistic, and analytical) have been implemented in Phaeaco in ways that simulate human performance, but do not emulate the human procedures, especially at the lower, hardwired level. In other words, no claim is made that Phaeaco accurately models a human BP solver, although there is a nonzero probability that it could pass successfully a Turing-test inspired "imitation game" in which Phaeaco's BP-solving performance could fool a judge, who would mistake Phaeaco for a person.

There are many important issues that have not yet been incorporated into the current implementation. However, none of these issues indicates an inherent limitation of the architecture. Future enhancements implemented within the existing architectural framework should be able to render solvable most (if not all) of the BP's listed in Appendix A.

CHAPTER TWELVE

Beyond Vision

An assessment of the implications of the present work is offered in this final chapter, with an emphasis on related philosophical issues.

12.1 On the primacy of vision

According to a well-known view in cognitive science, the most abstract and seemingly perception-free thoughts in human cognition are based on metaphors of visual perception. This does not mean that abstract thinking necessarily involves manipulation of visual images, but that it evolved from explicit, visual-only perception (e.g., Johnson, 1987; Lakoff and Johnson, 1980). The ample use of spatiotemporal analogies in language has been used in support of this view: a person is "above" somebody else in a ranked hierarchy, and scores "below" average in a test; a mood can be "high", and a profile "low"; people must get "over" an unfortunate event, and act "under" "pressure"; a "top" executive "reshuffles" a "cabinet" from the "bottom-up", and tries to make "ends" "meet" by "moving" a "meeting" to a later date; an "appointed" attorney makes a "sharp" argument "before" the "court"; the list is endless.

The above is compatible with Hofstadter's view of analogy being at the core of cognition (§8.4). Each of the above abstractions causes an unintended analogy between spatiotemporal and conceptual structures, which passes completely unnoticed, because it has become automated in colloquial language.

If metaphors, or analogies (in Lakoff's and Hofstadter's terms, respectively), are behind such figurative uses of words that originated as primitives from the visual world, one would expect that a cognitive architecture originally proposed to solve problems in vision should also be extensible to handle representations that are less literal, and to some degree more detached from vision. Indeed, some examples might serve to illustrate how Phaeaco's structures could be abstracted. Consider the following sentence:

Joe kissed Mary on the cheek

It conveys not a particular string of English words or Roman letters, but an idea. The sentence could have been given in Turkish (Yusuf Meryem'i yanağından öptü), in Greek (ο Ιωσήφ φίλησε τη Μαρία στο μάγουλο), in American Sign Language, or in any other system capable of expressing this thought. Suppose a cognitive system already knows the representations of concepts such as the individuals Joe and Mary (both instances of the concept "person"), an act of kissing, and how cheeks are related to people. How could these concepts be put together to form a single thought? Consider starting with something analogous but simpler, which we already know how to represent in Phaeaco's terms:

Line λ_1 touches line λ_2 at point P

This could be a sentence generated by a module that looks at the input shown in Figure 12.1 with the purpose of describing in words what it sees.

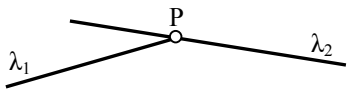


Figure 12.1: "Line λ_1 touches line λ_2 at point P"

If the previous input presents no representational challenge for Phaeaco, a slightly more challenging situation can now be considered (Figure 12.2).

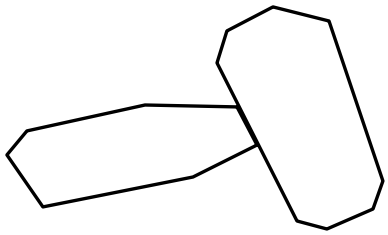


Figure 12.2: The relation "touches", abstracted slightly

The objects in Figure 12.2 appear to be related in an analogous way to that of Figure 12.1. The relation is slightly more complex (how much of the touching object is hidden under the touched one? If their common region is not a line but a point, should this information not be included in the representation?) but it can still be described by the word "touch". And it is possible to continually make the relation more complex.

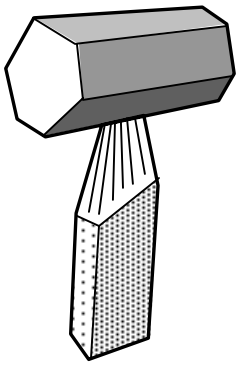


Figure 12.3: Is this the same relation?

The object in Figure 12.3 could be a familiar (albeit somewhat primitive) hammer, or it could be the depiction of an object supporting another one. In any case, we would probably feel comfortable using a new verb, such as "holds", or "supports", for the relation, and new nouns, such as "head" and "handle" for the constituent objects. But still there would be little doubt that, given suitable primitives, all these visual constructs would be analogous, and thus representable by means of the same principles. More complex representations would result from adding motion into the picture (e.g., the lower object chasing the upper one), or turning the objects to animated cartoons of animals, or people, who perform some complicated act, such as kissing. Finally, a useful (and natural in Phaeaco's architecture) next step would be to form a pattern out of many sightings of kissing acts, so that what is stored in memory is not any particular such act, but kissing *in general*. It would also be useful to have an "image generator" available that, given the kissing pattern, generates an approximate "motion picture" of the event, using two generic characters as actors. The cognitive system could mentally inspect this approximation, for example, for drawing conclusions in further thoughts.

But we are still in the visual world. Although the generic kissing pattern is not grounded in particular individuals, it is still possible for the image generator to create an approximation of the event, and even to dress it with arbitrary details. The final, decisive step in abstraction would be to match together a number of events in which an agent *does something to* another agent.[93] This would leave us with a mere transitive verb in linguistic terms, and a sentence of the form "X acts on Y". There would be no use for the image generator in this case (Figure 12.4).

---

[93] Notice that it is only a peculiarity of English that we are forced to use a specific preposition, such as "to". Many other languages are more lenient, using a generic preposition, or none at all. The most abstract version of this thought does not need to include details such as tense, place, manner, etc. It is only some languages that force the speaker to be specific.

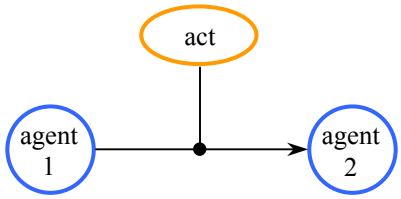


Figure 12.4: Abstraction for a transitive verb with two arguments

The only specificity in the representation of Figure 12.4 is that there are two agents, and one is acting on the other (the arrow has a direction). This was a result of abstracting the original visual relation, "touches". Similarly, it is possible to abstract a relation such as "meet", which is symmetric with respect to the participating agents, thus replacing the arrow in Figure 12.4 with an undirected line. An abstract K-point-like relation (§7.4.2) would be one that involves any number of agents. Some featural nodes (such as "red") would generalize to ideas usually expressed as adjectives, whereas other featural nodes (such as "tapers off") would be expressed by intransitive verbs. But note that this is a distinction that *some* languages (including the Indo-European family) force upon speakers;[94] if only abstract "mentalese" is concerned, most adjectives and intransitive verbs would correspond to Phaeaco's featural nodes.

Naturally, the above is an oversimplified sketch of the extremely complicated architectural tapestry of adult human cognition. Nonetheless, it serves to illustrate the point that even the most complex edifice built by nature and known to human cognition — namely, its own self — could be based on well-understood visual primitives, and principles that build upon them.

---

[94] There are languages, such as Japanese, that use verbal forms to express what in English would be described by a predicate (e.g., "is red").

12.2 Does Phaeaco "understand" anything?

As stated, the question in the title of this section is ill-posed. "To understand" is a concept that initially seems easy to grasp, but close examination reveals it to be less crisp than it first appears. To be able to give a thorough answer, after first agreeing on the meaning of the question, it is useful to review the misperception of "understanding" in John Searle's Chinese Room thought experiment.

12.2.1 In defense of half of Searle's Chinese Room

In 1980, Searle published a thought-provoking *Gedankenexperiment,* known ever since as "the Chinese Room experiment" (Searle, 1980). In it, he asked the reader to imagine Searle being locked in a room, from which scripts written in Chinese can be exchanged with the external world. Some of these scripts are called "stories", and others "questions", but Searle is unaware of that. All he has is Chinese symbols that mean nothing to him, plus a set of rules written in English, which of course he understands as a native speaker. The rules tell him how to combine the symbols together to produce new symbols, which he sends as output from the room. To the observers (native Chinese speakers), it appears as if there is a person in the room who understands Chinese just as well as they do. But in reality (says Searle) there is no one with an understanding of Chinese; it is only a "formal symbol manipulator" that achieves this. In contrast, if he were to be given questions in English, he would have no problem answering back in English, but this time he would have a full understanding of what he was being asked, so he would need no formal rules to explain to him how to manipulate the English symbols. In the case of English there is a mind at work; in the case of Chinese there is only a mindless automaton.

Searle proposed this thought experiment in support of a view that, by today's standards and hindsight, makes a lot of sense. A few years earlier, some AI researchers, most notably Roger Schank and his colleagues at Yale, had produced programs they claimed could understand the script of a typical exchange between a customer and the staff in a restaurant (Schank and Abelson, 1977). For example, here is how Searle describes these programs:

> [S]uppose you are given the following story: "A man went into a restaurant and ordered a hamburger. When the hamburger arrived it was burned to a crisp, and the man stormed out of the restaurant angrily, without paying for the hamburger or leaving a tip." Now, if you are asked "Did the man eat the hamburger?" you will presumably answer, "No, he did not." Similarly, if you are given the following story: "A man went into a restaurant and ordered a hamburger; when the hamburger came he was very pleased with it; and as he left the restaurant he gave the waitress a large tip before paying his bill," and you are asked the question, "Did the man eat the hamburger?" you will presumably answer, "Yes, he ate the hamburger." Now Schank's machines can similarly answer questions about restaurants in this fashion.

Searle then went on to claim that such programs have no understanding of the situation whatsoever, and to support his claim he proposed the Chinese Room experiment. He said these programs are mere symbol manipulators: they receive symbols from the external world (a passage in English that describes the story), of which they have no understanding, because those symbols are not connected to anything material (e.g., real hamburgers), just like his Chinese symbols are

disconnected from the world; those symbols are manipulated by means of formal instructions (the program); and more symbols are outputted, still not understood.

To a certain extent, Searle's argument is valid. Early AI programs attempted to handle *too much* — much more than was justified by the simplistic theories and meager computing resources of the time. For example, a real hamburger is too complex an object to be represented by a "100% fat-free Lisp atom"[95] such as "hamburger". There is no meaning in a string of letters. Even if this string is connected to other strings, such as "bread bun", "beef patty", "tomato", and so on, the whole thing is a pathetic caricature of a representation for a real hamburger, let alone customers, waitresses, tips, bills, restaurants, etc.

12.2.2 But the other half of the room is empty

Had Searle restricted his critique to Schank's restaurant scripts, there would be little against which to argue. Unfortunately, Searle extended his argument to include *any* program: anything non-human that engages in information-processing activity. He rejected even neuronal functionalism (the neuron-for-neuron replacement by devices functionally identical to the neurons they are replacing). The only machinery he granted might actually engage in true thinking was a molecule-for-molecule replica of the human brain. In other words, if the machine is not made of "the right stuff", it cannot think.

Searle's argument has caused strong reactions, including excellent counter-arguments and discussion (see Hofstadter and Dennett, 1981). It is unnecessary to repeat those arguments here. What is relevant is Searle's conception of "understanding". The following is his reaction to the suggestion that "understanding" might not be a two-valued predicate:

---

[95] Hofstadter's expression.

> In many […] discussions one finds a lot of fancy footwork about the word "understanding". My critics point out that there are many different degrees of understanding; that "understanding" is not a simple two-place predicate; that there are even different kinds and levels of understanding […]; that in many cases it is a matter for decision and not a simple matter of fact whether *x* understands *y;* and so on. To all of these points I want to say: of course, of course.

But in the rest of his article he shows that he does not accept even the tiniest bit of such variation in understanding unless the device is made out of the right stuff: "biological (i.e., chemical and physical) structure [which] is causally capable of producing perception, action, understanding, learning, and other intentional phenomena." What phenomena are "intentional"? What is intentionality?

> Intentionality is by definition that feature of certain mental states by which they are directed at or about objects and states of affairs in the world. Thus, beliefs, desires, and intentions are intentional states; undirected forms of anxiety and depression are not.

Given the above definition, one might think that Searle could allow a system like Phaeaco to have intentionality, because Phaeaco has mental states. And yet, no. Phaeaco is not made of the right stuff, and thus is not "causally capable" of producing perception, etc.

This line of reasoning, which amounts to no more than a philosopher's stubborn pontification of what is and is not allowed to be called "mental", evokes little patience among most cognitive scientists. For it is *unscientific* to decide by decree what is and is not mental, or which type of system is allowed to possess

intentionality and which not. Such an attitude introduces an arbiter's subjective bias, and it is a violation of scientific objectivity to accept an arbiter's decision for answering questions. Scientifically valid answers must be empirical. But, more fundamentally, the questions themselves (whether some states are "mental", whether a system is "intentional", etc.) are unscientific pseudo-questions, since it is not possible to answer them in a clear-cut, yes/no manner: there is an entire spectrum of mentality, intentionality, intelligence, and consciousness, which ranges from the absolute zero of a stone, to the highest degree known to us, the adult human mind. An elucidation of this range in all aspects of cognition is attempted next.

12.2.3 On the inadequacy of classical logic in cognition

Traditions that have been around for thousands of years ought not be abandoned lightly, particularly if they have proved useful in our quest to understand and explain our environment. Such a tradition is Boolean logic, which, although it took its name from George Boole, a 19th century British mathematician, has its roots in Aristotle's *Prior Analytics* (Aristotle, 1992)[96], and was developed further by Augustus De Morgan, Gottlob Frege, Bertrand Russell, Kurt Gödel, and Alfred Tarski, to name only a few with landmark publications. Traditional Boolean logic has been enriched with variants that can handle possibility and necessity (modal logics), and others that deal with uncertain set membership and approximate reasoning (fuzzy logic). Nonetheless, the dominant mode of thinking among philosophers (and almost always among laypeople) is that of traditional Boolean logic. For example, an agent either has a mind or not; has intentionality or not; is

[96] Note, however, that the famous syllogism concluding "Socrates is mortal" is not found in this work. Aristotle used letters to denote abstract syllogisms; the famous one about Socrates is from Sextus Empiricus (2nd – 3rd C. AD), an Alexandrian and Athenian physician and philosopher.

conscious or unconscious; and so on. Those who study the philosophy of mind are often entrapped in this mode of reasoning, and allow no room for gray regions of uncertainty.[97] To see that this framework of thought is unproductive, consider the example of trying to ascribe the concept of *volition* to various agents:

- Is a person who suddenly feels a strong need for a cup of coffee, and is now preparing the coffee maker, etc., *willing* to have that coffee? Most people would answer, obviously yes. How else could it be?
- Is a baby who cries, feeling hunger for milk, willing to have milk? Again most people would answer yes, unwilling to admit less volition to a member of our own species, although it is not hard to argue that an adult's concept of hot coffee is much richer that a baby's concept of milk. In Phaeaco's terms, the conceptual representation of coffee, reached in the adult's LTM, would be more complex than the baby's representation of milk in an easily quantifiable way.
- Is a pet dog scratching on the front door displaying volition? Does the dog want to go out for an afternoon walk? Pet lovers would answer with an emphatic "Yes!" But they would agree that their own concepts are more complex than those of their dog. People who are not pet lovers, after overcoming their bias in favor of our species, might agree that an adult dog's concepts are more complex than those of a newborn human.
- Does a goldfish that opens its mouth to swallow a chunk of food want to eat it? Here the notion starts becoming murky. What does a goldfish

[97] This is quite understandable, because one of the unifying themes of Western thought has been the urge towards absolute clarity and precision. Without this urge, almost certainly none of the so-called "exact sciences" (astronomy, physics, chemistry, etc.) would exist.

know about "food"? Its tiny brain does not appear designed for entertaining such a concept, because it is not necessary for its survival: all the fish needs to do is *react* to a floating object of suitable size by opening its mouth.

- Does an amoeba want to engulf a tiny piece of algae in the vacuole that it just created out of two pseudopodia? "Certainly not" is the first reaction that comes to mind. Amoebas do not have neurons with which to think; they are unicellular organisms. But the way an amoeba senses its environment with its pseudopodia, the way it stops in front of "food" and creates the vacuole exactly where it must be made to engulf the food, are all actions strongly reminiscent of volition. Does a living thing need neurons to act as if it has the rudiments of volition?
- Consider also the hydras, jellyfish, corals, anemones, and other members of the phylum Cnidaria. These animals usually have neurons, but do not move around in search of their food. Food comes to them.
- Does a molecule of benzene that just lost an atom of hydrogen want to replace it with another atom of hydrogen (or chlorine, etc.)?
- Does a magnet want to stick on a refrigerator?

Some of these suggestions might appear absurd. But in their entirety they serve to illustrate the range, and perhaps the origins, of the concept "volition". What is more absurd is to attempt to draw a line beyond which volition (and hence, consciousness) does not exist. Would it be reasonable to draw this line between our species and anything else? Before there were *Homo sapiens* there were *Homo erectus*, and there was no "first" *H. sapiens*, only a smooth transition. Every attempt to define the first conscious individual is arbitrary, unscientific, and vacuous. The same is true of any attempt to define the first creature with mental

states, intentionality, and many other notions that are given center-stage roles in the philosophy of mind.

If it is conceivable to see an *Amoeba proteus* as nothing more than a chemical (or biological) machine with the rudiments of volition, it is also conceivable to assign non-zero cognitive attributes to a computational machine, such as Phaeaco, which was designed to entertain representations. Phaeaco may not "want" to solve a BP as soon as it sees one any more than a spider "wants" to consume its prey, but it can look at the BP, construct representations of its contents, generalize them to visual patterns, activate concepts in its LTM, and "decide" to look again, and again if needed, at various parts of its input. How many animal species with sensing devices made of the "right stuff" can be claimed to build representations of what they perceive?

12.3 On the inner "I" and related issues

The discussion in §12.2.3 leads to the conclusion that every complex notion in cognition, including consciousness and the sense of self, corresponds not to a simplistic two-valued traditional predicate, but either to a range of continuous values, or to a multidimensional space. Whichever is the case, every complex cognitive notion has evolutionary origins. This makes it possible for cognitive scientists to examine "simpler" versions, much like biologists who, in order to study the complex human eye, extend their investigation to include eyes in the entire evolutionary spectrum, starting from simple pigmented light-sensitive spots on single-celled animals (Dawkins, 1996, p. 85). If biologists benefit by studying in this way the eye, cognitive scientists might also benefit by similarly studying the "I".

12.3.1 What it would take for Phaeaco to possess an "I"

As it stands in its current implementation, Phaeaco has no "I": in a range from 0 to 1 (with the value 1 assigned to human cognition), Phaeaco would be assigned exactly 0. But the present chapter is not about how Phaeaco is, but about how it could improve and appear to be. What would it take to develop a sense of "I"?

First note that the BP domain does not appear to be sufficiently conducive to developing such a sense. The domain includes objects (BP's) that can be examined without reference to the agent solving them, or to other agents posing them. Even if such other agents were included (and a dialogue went on between solver and problem-poser), the relation between such agents and objects of the domain would be at best very tenuous and forced.

A different domain that involves agents-with-an-"I" (conscious agents) participating actively and necessarily in the domain would be more helpful. Game-playing, particularly of the sort that involves an understanding of the opponent's role and psychology (e.g., poker) might seem more fruitful than the BP-domain for this purpose.

Assuming a suitable domain for experimentation is identified, Phaeaco should start with a single node representing "I", devoid of any connections, which would be as useful a representation as the node for a single object that has made no connections yet; in other words, its utility would be nil. But each event that involved Phaeaco, either as the subject or the recipient of an action, would be recorded and linked to this self-node. Similar nodes for other agents should exist as well. Eventually an event-ful ("episodic"), very complex memory would develop around the self-node. The system should also possess a psychologically plausible sense of real time, because a large part of our sense of "who we are" is related to a correct chronological ordering of events in our memories.

There would be no single moment in this build-up of the self-node in which the system would suddenly become conscious. The degree of consciousness would be correlated with the complexity of the self-node, and the clarity with which it would be capable of observing itself, introspectively.

In addition, philosophers would feel free to speculate on whether this future, imagined version of Phaeaco is conscious, but their opinions would be irrelevant because their minds are not made of the "right stuff". Only Phaeaco would be able to answer authoritatively this question.

12.3.2 Can Phaeaco have subjective experiences ("qualia")?

"Subjective experiences" is another favorite topic in the philosophy of mind, but cognitive scientists generally consider it ill-defined. What is interesting in this case is that programs like Phaeaco make it possible to explore these questions from a different, "hands-on" perspective. Phaeaco appears to be replicating (roughly) the perception of an agent with subjective experiences (assuming we are willing to admit the sensibility of the notion): photons from the external world can hit a camera (corresponding to the lens of an eye), be directed to the computer's screen (corresponding to the retina), and thereby analyzed by Phaeaco, building representations, and performing complex actions that can only be performed by people (e.g., coming up with the correct word that describes what exists in the world). So, could Phaeaco be said to have subjective experiences?

Before attempting to answer this question, the notion of "subjective experience", or "qualia",[98] must be clarified. It is not easy to define qualia, because they are a quintessentially subjective notion (definitions are usually understood to be objective), so it is easier to describe them by examples. When a

[98] Philosophers use the shorter term "qualia" (Latin for "qualities"; singular: "quale") instead of "subjective experiences", and this term will be adopted hereafter for brevity.

person experiences the redness of a rose, the smell of burnt rubber, the taste of licorice, the sound of a dog barking, the pain from having the skin punctured with a sharp object, the pain of losing a loved one, in every such case the person is having a mental state that can have both an objective and a subjective ("first-person") description. The subjective description, obtained introspectively, is the "quale".

Some philosophers claim that qualia are irreducible, non-physical entities that must be added to the ontology of physics, on a par with mass, energy, time, and space, in order to obtain a complete view of the world (Chalmers, 1996). Others think that qualia are first-person illusions: there is nothing in a subjective experience that cannot also be described objectively (Dennett, 1991, p. 372).

The following thought experiment is sometimes presented as a decisive argument for the existence of qualia:[99]

> Mary suffers congenitally from complete achromatopsia, i.e., she can see no colors. She only sees shades of gray, as in black-and-white photographs. She becomes a brilliant neuroscientist, and is capable of knowing all that there is to know about color vision. Technological advancements allow neuroscientists to examine the human brain as it experiences anything in complete detail, down to the last synapse, so Mary has 100% knowledge of what happens to the brain of a normal person while perceiving a colored scene. Mary likes to visit a colorful garden with flowers (which appear to her in shades of gray), examining her own brain. One day, progress in neuroscience makes it possible to correct whatever was

[99] Originally proposed by Frank Jackson (Jackson, 1982), but simplified here in some technical details.

> damaged in Mary's vision, and now she can experience colors. She goes to the same colorful garden, examines her brain, and notices a different pattern of neuronal activation. But, looking at the flowers directly, she exclaims: "So, *that* is what it is like to see colors!"

This and similar thought experiments[100] are used by proponents of the notion of "irreducible qualia" to claim that although Mary comes to know all there is to know about color perception, she still experiences something new when she sees the flowers in color for the first time (Tye, 1986).

But Daniel Dennett has argued that if Mary is assumed to be "omniscient" regarding color vision, then her surprise upon seeing in colors is unjustified: she already knows how her brain will react, and she can even predict the feelings that the colored flowers will give to her, having seen the neural correlates of such feelings in brains of other people. The reader of this experiment, Dennett argues, is tricked into "not following directions" (Dennett, 1991, p. 399):

> The reason no one follows directions is because what they ask you to imagine is so preposterously immense, you can't even try. The crucial premise is that "She has *all* the physical information." That is not readily imaginable, so no one bothers. They just imagine that she knows lots and lots — perhaps they imagine that she knows everything that anyone knows *today* about the neurophysiology of color vision. But that's just a drop in the bucket, and it's not surprising that Mary would learn something if *that* were all she knew.

---

[100] In Jackson's original formulation, Mary has normal vision but is raised as a captive in a room where everything is black-and-white, she is given books and a TV, there are no mirrors (or her skin is covered in gray paint), and is released one day by her captors into the colorful garden.

Possibly the essence of this contention can be better understood if we examine Mary's representation of a colored object in her "direct" experience (after she is cured), and contrast it with her representation of the "indirect" experience.

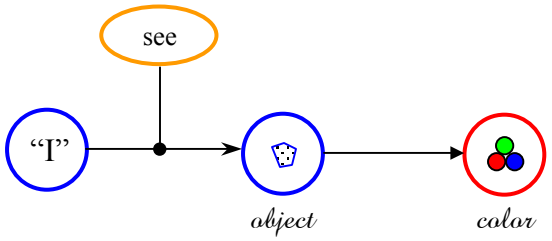


Figure 12.5: Representation of "I see a colored object"

Figure 12.5 shows a simplification of what Phaeaco could construct as a representation of the idea "I see a colored object". The feelings caused by the sighting should also be part of the figure.

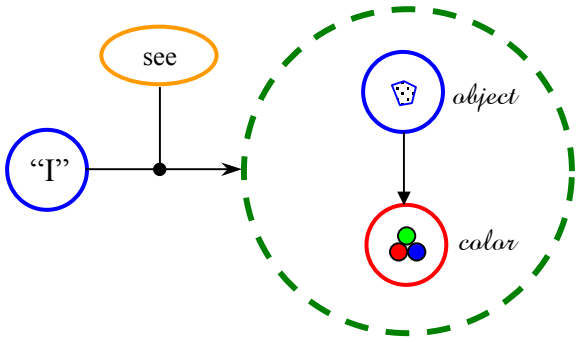


Figure 12.6: Representation of "I see my representation of a colored object"

In contrast, Figure 12.6 is a representation of the thought "I see everything that is created in my memory as a result of seeing a colored object". The dashed circle around the representation is supposed to be encompassing everything modified in Phaeaco's Workspace and LTM as a result of experiencing the colored object, including the feelings associated with the experience, and possibly

even the "I see" part (not shown in the figure). The difference is that this time what is seen is not an object causing some feelings, but a representation of the object (plus the feelings and everything else). This is what Mary, the expert neuroscientist, would "see" by examining her internal states.

The above, of course, is only a rough sketch, a caricature of what might occur in reality. But it helps to depict the difference between the two views. It is clear, for example, that there is a difference between the two views (all philosophers agree that there is a difference). It also shows that the first view does not miss any ethereal, non-physical qualia (there are no qualia); it is a representation that is not examined from the outside, from a third-person perspective. The second view is precisely a third-person perspective, but, agreeing with Dennett, it does not miss anything that exists in the first view; it is simply a different, somewhat detached perspective, and that is the source of the philosophical contention.

12.4 Summary

Vision, the most essential of all senses in primates, forms a fundamental platform upon which our higher cognitive abilities evolved. Such abilities include our sense of time, consciousness, self, and even our illusory first-person perspective of the world.

12.5 A recapitulation of ideas introduced in this thesis

In conclusion, the present work introduced, among others, the following notions:

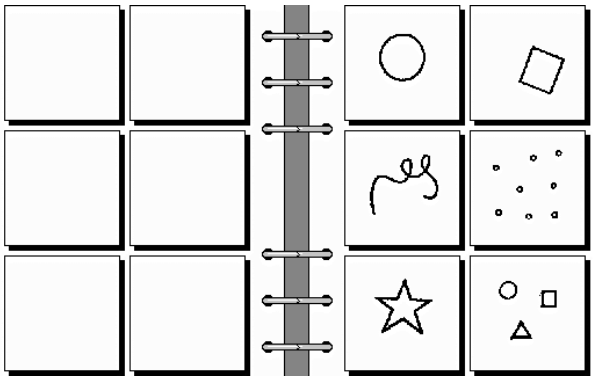
- Visual representations (chapter 7): a number of visual primitives can be combined in a principled way to build structures that represent the input.

- The use of statistics (chapter 8) for computing a psychological difference between instances of input, and for creating summary representations ("visual patterns") of sufficiently similar input instances.
- A novel algorithm for the formation of groups of representations (§8.3.2).
- An LTM (chapter 9) comprising interconnected concepts, which are visual patterns that migrated into the LTM; and a practical indexing scheme (indispensable in serial computers) for accessing the LTM concepts.
- An improvement over the notion of "concepts coming closer together", as used in previous FARG architectures, that leads to more reliable long-term learning (§9.3.1).
- A proposal for how time can be used for learning from positive only examples without resulting in overgeneralizations (§9.4.2), which implies that forgetting is not necessarily an undesirable property of cognition.
- The separation of processing into two parts: a computationally intensive and cognitively inaccessible ("retinal") level and a more abstract but accessible ("cognitive") level, where "accessible" means "what the system could think and talk about" if it possessed some degree of consciousness and were equipped with language.
- A vision of convergence between natural and programmed cognition (§4.3, Figure 4.9), acknowledging the fact that the two are based on hardware elements with radically different properties, but attaining "mind" as a common goal.

It is hoped that these ideas, and others introduced in the present thesis, will be useful for future work in the automation of cognition.

Appendix A: Bongard Problems

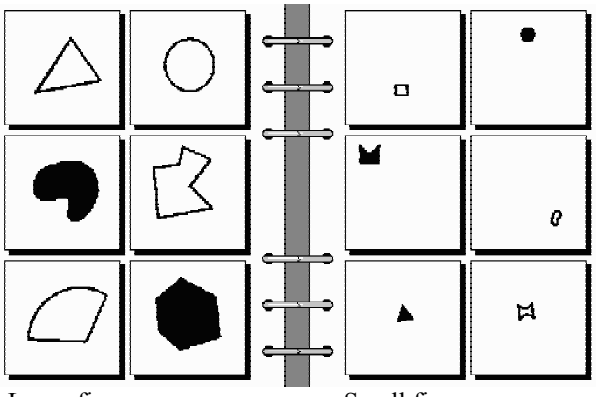
The 200 BP's mentioned in the main text are given below. The first 100 were designed by Bongard, the next 56 by Hofstadter, and the last 44 by the author. The results of the experiment described in §3.2 are given in a table next to each BP. The number of correct answers, average time, and standard deviation are shown. All times are in seconds. Also shown are: the number of incorrect answers, their average time, the number of wrong answers, their average time, and the total number of subjects (sample size). Wherever available, Phaeaco's corresponding performance values are shown, from a sample of 100 runs.



Empty picture Not empty picture

BP #1

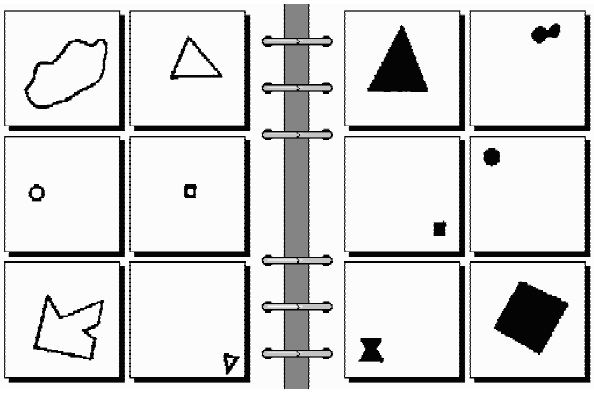
Performance	*Human*	*Phaeaco*
Correct	31	100
Avg. time	*6.9*	*7.0*
Std. dev.	*5.1*	*0.4*
Incorrect	0	0
Avg. time		
No answer	0	0
Avg. time		
Sample size	31	100



Large figures Small figures

BP #2

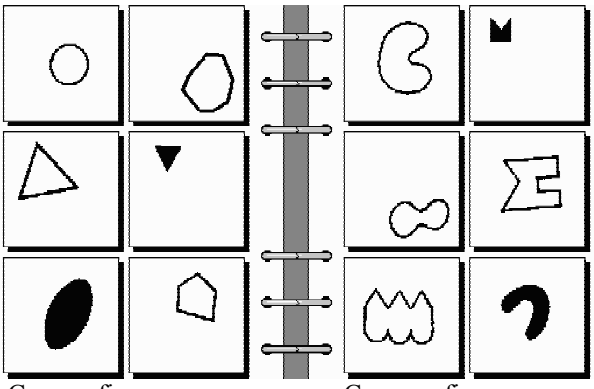
Performance	*Human*	*Phaeaco*
Correct	28	82
Avg. time	*13.6*	*11.2*
Std. dev.	*9.6*	*2.5*
Incorrect	0	18
Avg. time		*13.4*
No answer	0	0
Avg. time		
Sample size	28	100



Outlined figures Filled figures

BP #3

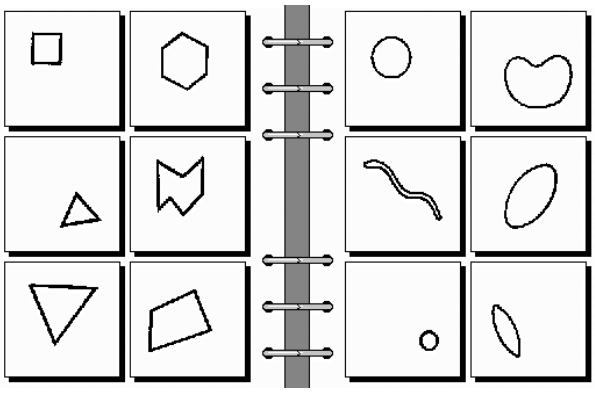
Performance:	*Human*	*Phaeaco*
Correct	28	99
Avg. time	*7.9*	*4.3*
Std. dev.	*6.1*	*0.2*
Incorrect	0	1
Avg. time		*8.0*
No answer	0	0
Avg. time		
Sample size	28	100



Convex figures Concave figures

BP #4

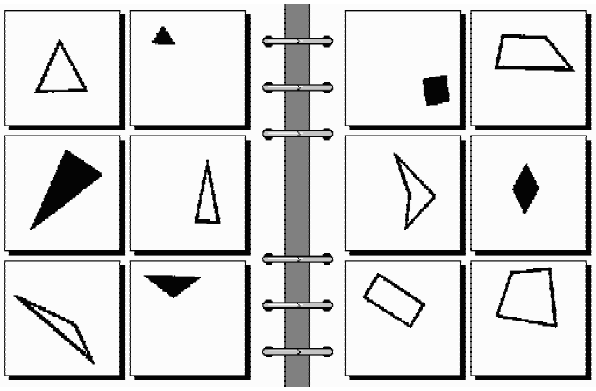
Performance:	*Human*	*Phaeaco*
Correct	5	20
Avg. time	*17.6*	*8.5*
Std. dev.	*10.9*	*0.3*
Incorrect	9	3
Avg. time	*30.4*	*9.3*
No answer	17	77
Avg. time	*37.8*	*23.8*
Sample size	31	100



Polygons Curvilinear figures

BP #5

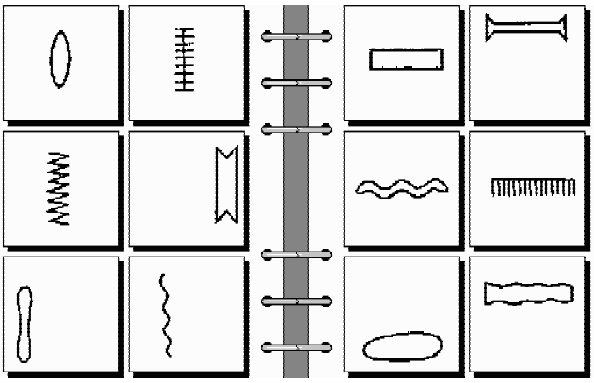
Performance:	*Human*	*Phaeaco*
Correct	28	60
Avg. time	*11.0*	*8.9*
Std. dev.	*9.1*	*2.8*
Incorrect	2	18
Avg. time	*5.0*	*16.6*
No answer	1	22
Avg. time	*16.0*	*21.5*
Sample size	31	100



Triangles Quadrilaterals

BP #6

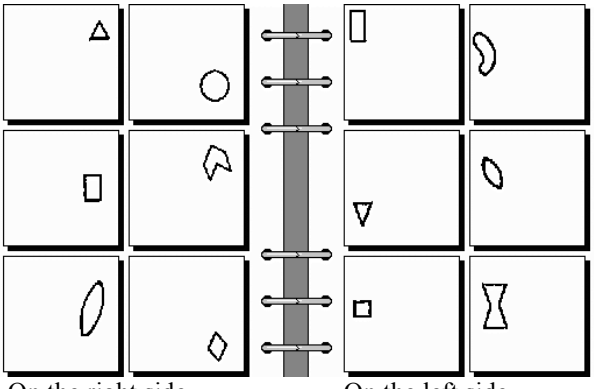
Performance:	*Human*	*Phaeaco*
Correct	26	70
Avg. time	*18.8*	*4.4*
Std. dev.	*16.3*	*3.8*
Incorrect	2	11
Avg. time	*26.5*	*12.5*
No answer	3	19
Avg. time	*27.7*	*14.6*
Sample size	31	100



Elongated vertically Elongated horizontally

BP #7

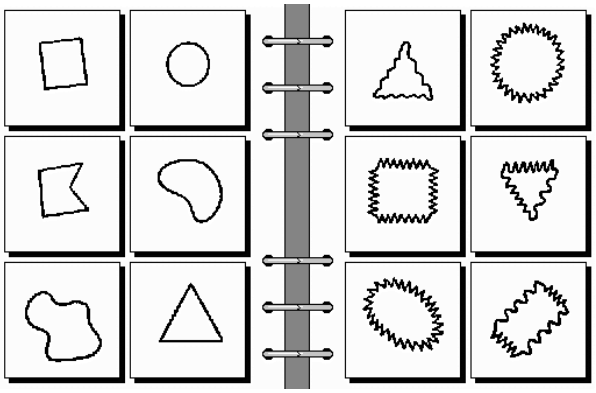
Performance:	*Human*	*Phaeaco*
Correct	27	
Avg. time	*11.5*	
Std. dev.	*12.3*	
Incorrect	0	
Avg. time		
No answer	3	
Avg. time	*25.7*	
Sample size	30	



On the right side On the left side

BP #8

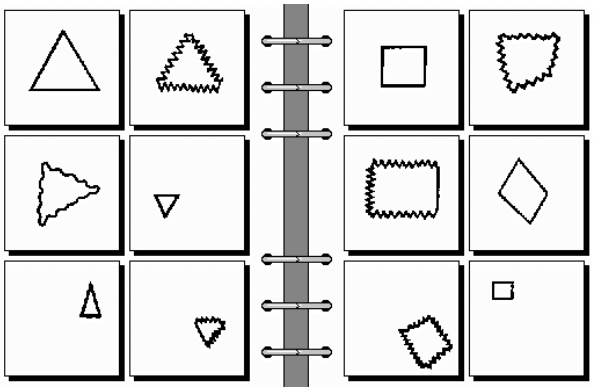
Performance:	*Human*	*Phaeaco*
Correct	24	23
Avg. time	*21.0*	*8.2*
Std. dev.	*20.9*	*0.5*
Incorrect	0	4
Avg. time		*15.4*
No answer	7	73
Avg. time	*23.0*	*18.2*
Sample size	31	100



Smooth contours Wiggly contours

BP #9

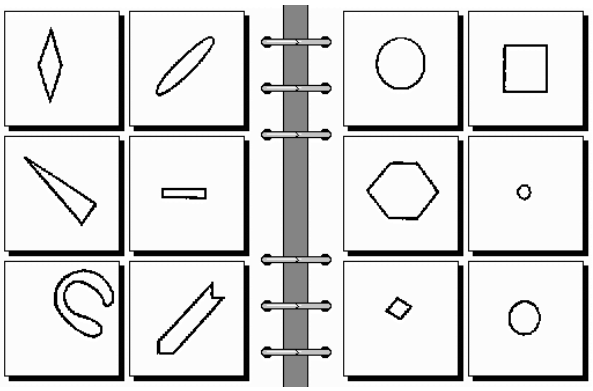
Performance:	*Human*	*Phaeaco*
Correct	31	
Avg. time	*10.5*	
Std. dev.	*7.3*	
Incorrect	0	
Avg. time		
No answer	0	
Avg. time		
Sample size	31	



Triangles Quadrilaterals

BP #10

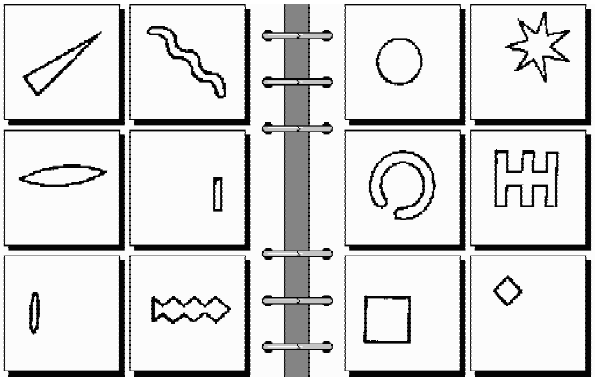
Performance:	*Human*	*Phaeaco*
Correct	27	
Avg. time	*12.2*	
Std. dev.	*9.7*	
Incorrect	2	
Avg. time	*27.0*	
No answer	2	
Avg. time	*26.5*	
Sample size	31	



Elongated figures Not elongated figures

BP #11

Performance:	*Human*	*Phaeaco*
Correct	15	83
Avg. time	*23.7*	*16.3*
Std. dev.	*13.7*	*6.4*
Incorrect	6	4
Avg. time	*38.0*	*21.3*
No answer	10	13
Avg. time	*30.6*	*23.7*
Sample size	31	100

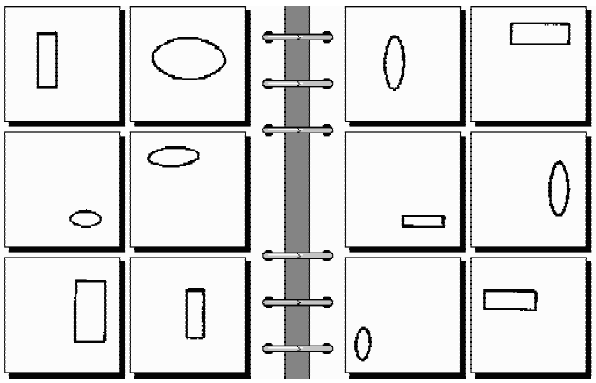


Elongated convex hull

Not elongated convex hull

BP #12

Performance:	*Human*	*Phaeaco*
Correct	7	
Avg. time	*33.1*	
Std. dev.	*14.4*	
Incorrect	2	
Avg. time	*21.5*	
No answer	21	
Avg. time	*30.5*	
Sample size	30	

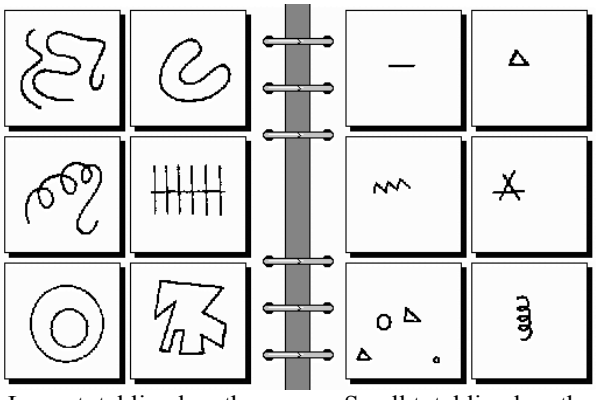


Vertical rectangles or horizontal ellipses

Horizontal rectangles or vertical ellipses

BP #13

Performance:	*Human*	*Phaeaco*
Correct	19	
Avg. time	*15.5*	
Std. dev.	*7.0*	
Incorrect	2	
Avg. time	*9.5*	
No answer	2	
Avg. time	*14.0*	
Sample size	23	

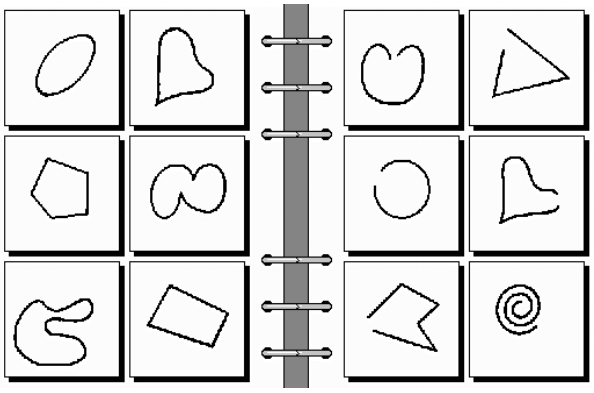


Large total line length

Small total line length

BP #14

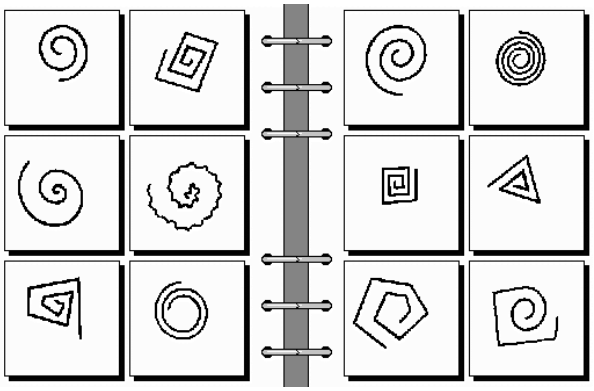
Performance:	*Human*	*Phaeaco*
Correct	24	
Avg. time	*15.0*	
Std. dev.	*12.3*	
Incorrect	0	
Avg. time		
No answer	1	
Avg. time	*1.0*	
Sample size	25	



Closed regions Open regions

BP #15

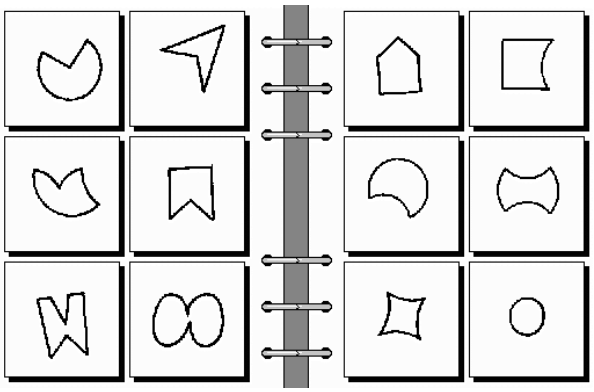
Performance:	*Human*	*Phaeaco*
Correct	27	99
Avg. time	*7.7*	*7.6*
Std. dev.	*3.6*	*0.3*
Incorrect	1	0
Avg. time	*7.0*	
No answer	3	1
Avg. time	*30.3*	*22.8*
Sample size	31	100



Counterclockwise spiral Clockwise spiral

BP #16

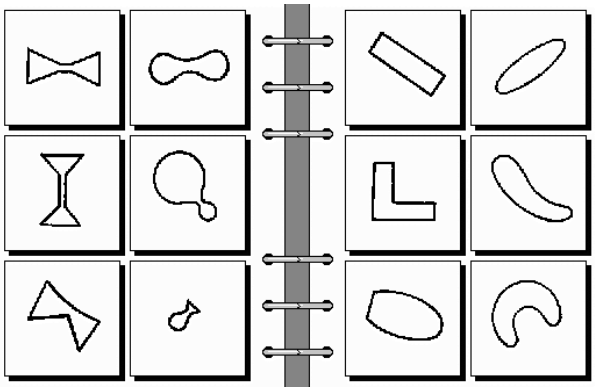
Performance:	*Human*	*Phaeaco*
Correct	9	
Avg. time	*29.4*	
Std. dev.	*20.2*	
Incorrect	5	
Avg. time	*27.4*	
No answer	10	
Avg. time	*21.6*	
Sample size	24	



Angle directed inward No inward angle

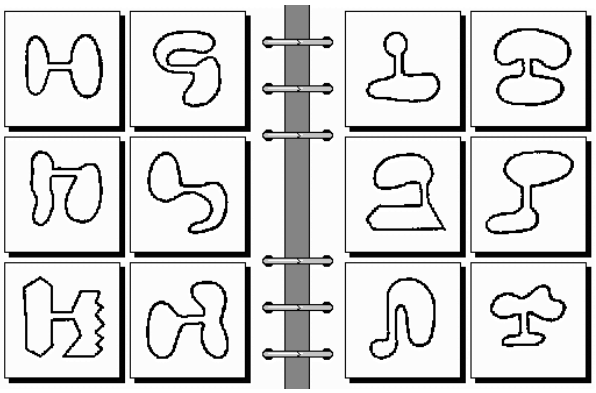
BP #17

Performance:	*Human*	*Phaeaco*
Correct	6	
Avg. time	*35.5*	
Std. dev.	*23.0*	
Incorrect	3	
Avg. time	*15.7*	
No answer	14	
Avg. time	*12.9*	
Sample size	23	



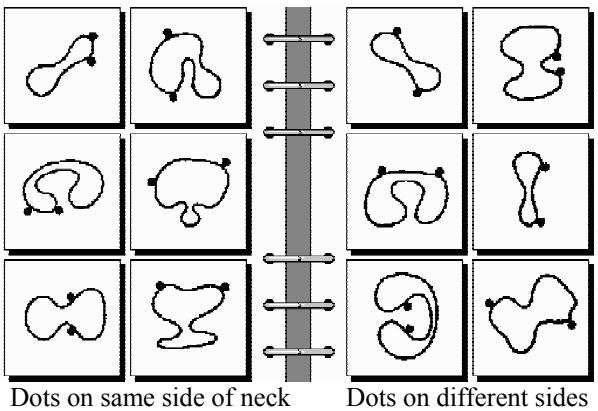
BP #18

Performance:	*Human*	*Phaeaco*
Correct	8	
Avg. time	*16.1*	
Std. dev.	*6.5*	
Incorrect	2	
Avg. time	*27.0*	
No answer	13	
Avg. time	*13.2*	
Sample size	23	



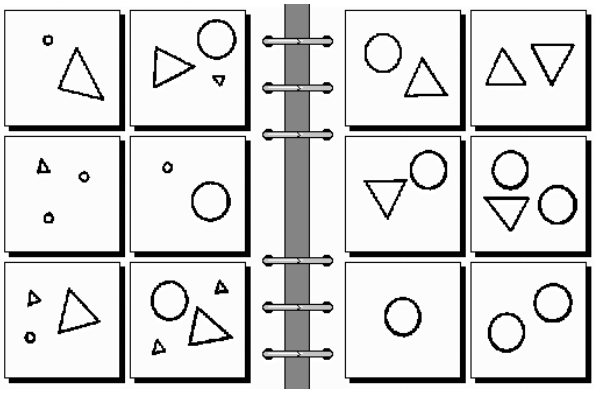
BP #19

Performance:	*Human*	*Phaeaco*
Correct	10	
Avg. time	*18.7*	
Std. dev.	*10.7*	
Incorrect	0	
Avg. time		
No answer	12	
Avg. time	*12.1*	
Sample size	22	



BP #20

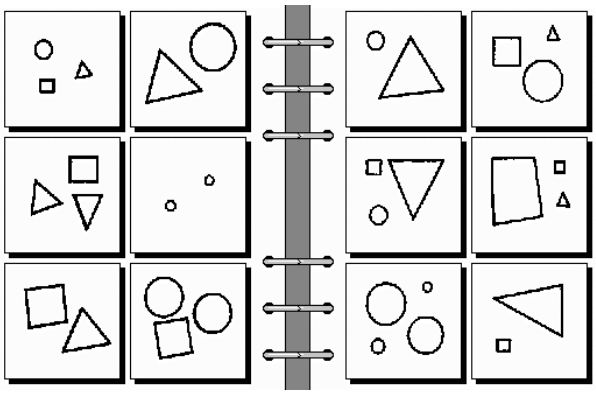
Performance:	*Human*	*Phaeaco*
Correct	5	
Avg. time	*21.4*	
Std. dev.	*6.2*	
Incorrect	2	
Avg. time	*26.0*	
No answer	15	
Avg. time	*16.3*	
Sample size	22	



Small figure present No small figure present

BP #21

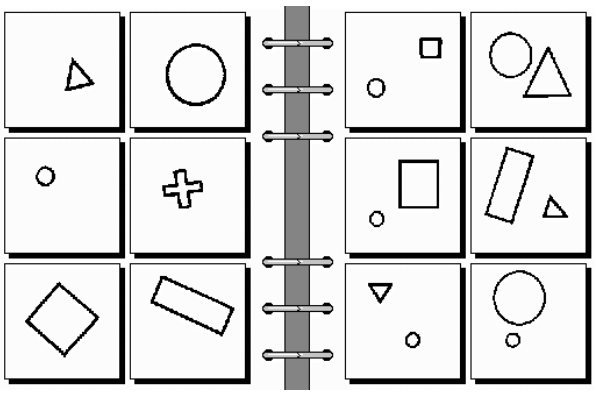
Performance:	*Human*	*Phaeaco*
Correct	20	34
Avg. time	*25.1*	*16.6*
Std. dev.	*16.3*	*3.4*
Incorrect	9	20
Avg. time	*31.4*	*43.4*
No answer	2	46
Avg. time	*42.5*	*40.1*
Sample size	31	100



Areas approximately equal Areas differ greatly

BP #22

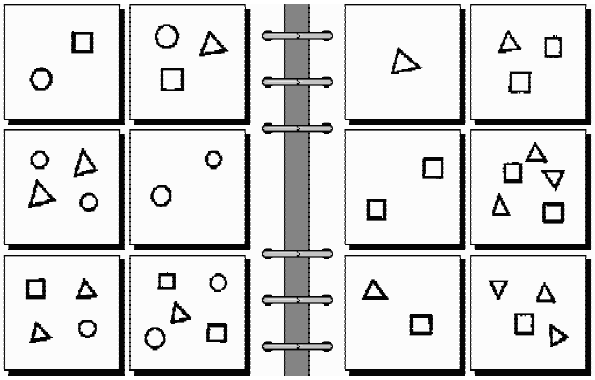
Performance:	*Human*	*Phaeaco*
Correct	11	13
Avg. time	*23.9*	*28.4*
Std. dev.	*20.2*	*7.2*
Incorrect	3	6
Avg. time	*42.0*	*24.0*
No answer	16	81
Avg. time	*35.0*	*23.3*
Sample size	30	100



One figure Two figures

BP #23

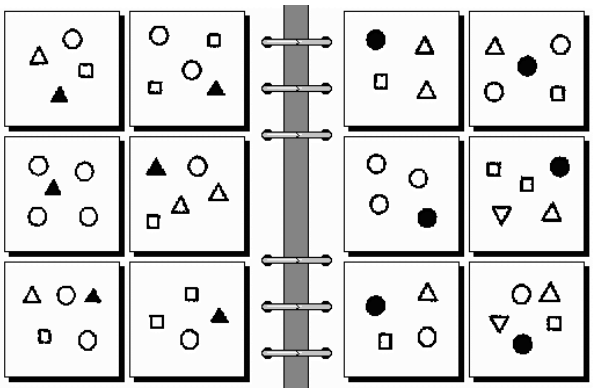
Performance:	*Human*	*Phaeaco*
Correct	30	83
Avg. time	*9.4*	*8.5*
Std. dev.	*5.4*	*2.5*
Incorrect	1	4
Avg. time	*20.0*	*19.5*
No answer	0	13
Avg. time		*22.5*
Sample size	31	100



Circle present No circle

BP #24

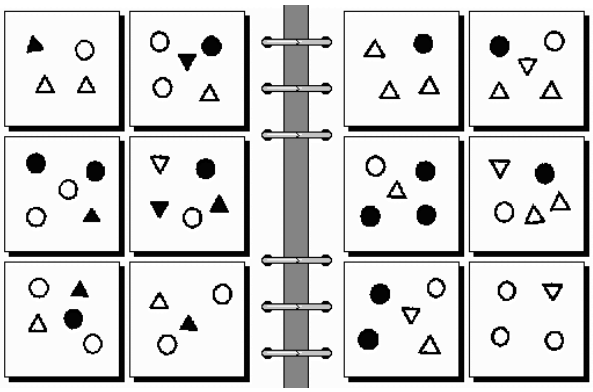
Performance:	*Human*	*Phaeaco*
Correct	20	
Avg. time	*21.0*	
Std. dev.	*12.2*	
Incorrect	3	
Avg. time	*20.0*	
No answer	7	
Avg. time	*31.7*	
Sample size	30	



Filled figure is triangle Filled figure is circle

BP #25

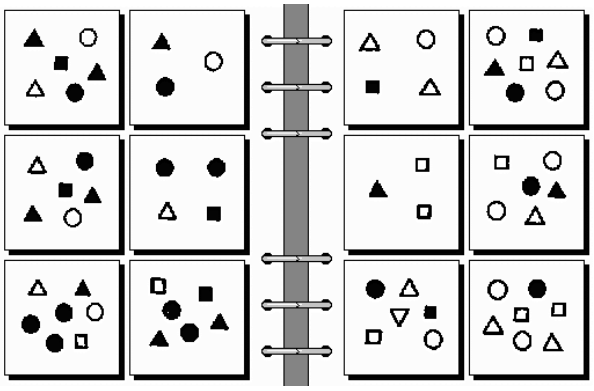
Performance:	*Human*	*Phaeaco*
Correct	22	
Avg. time	*9.7*	
Std. dev.	*6.9*	
Incorrect	0	
Avg. time		
No answer	4	
Avg. time	*5.3*	
Sample size	26	



Filled triangle exists No filled triangle

BP #26

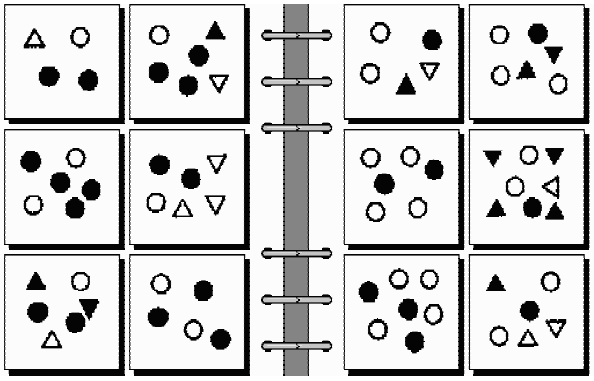
Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*15.7*	
Std. dev.	*10.1*	
Incorrect	4	
Avg. time	*17.0*	
No answer	5	
Avg. time	*13.6*	
Sample size	24	



More filled figures | More outlined figures

BP #27

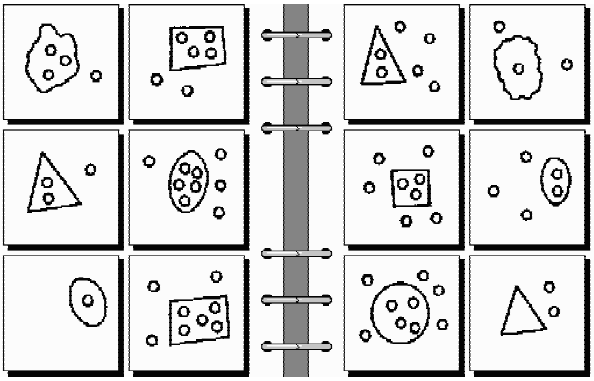
Performance:	*Human*	*Phaeaco*
Correct	8	
Avg. time	*18.1*	
Std. dev.	*16.0*	
Incorrect	4	
Avg. time	*50.75*	
No answer	10	
Avg. time	*15.8*	
Sample size	22	



More filled circles than outlined circles | More outlined circles than filled circles

BP #28

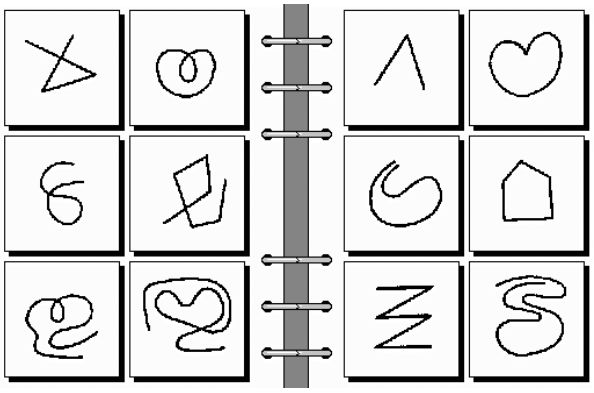
Performance:	*Human*	*Phaeaco*
Correct	0	
Avg. time		
Std. dev.		
Incorrect	0	
Avg. time		
No answer	21	
Avg. time	*20.5*	
Sample size	21	



More circles inside than outside | Fewer circles inside than outside

BP #29

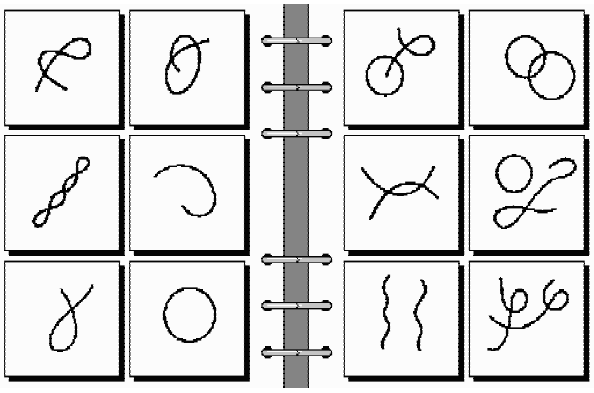
Performance:	*Human*	*Phaeaco*
Correct	3	
Avg. time	*23.7*	
Std. dev.	*4.1*	
Incorrect	11	
Avg. time	*24.5*	
No answer	11	
Avg. time	*29.3*	
Sample size	25	



Self-crossing line No self-crossing line

BP #30

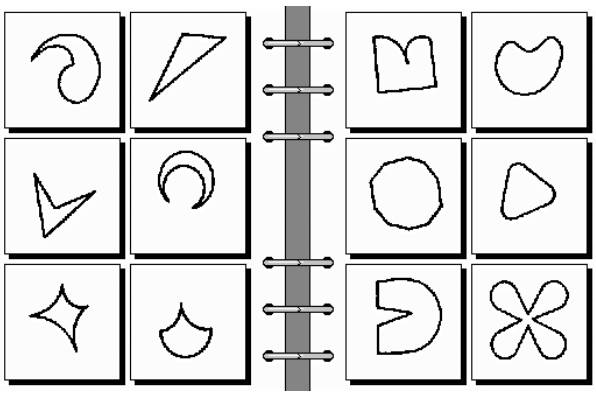
Performance:	*Human*	*Phaeaco*
Correct	7	
Avg. time	*19.4*	
Std. dev.	*9.9*	
Incorrect	6	
Avg. time	*23.8*	
No answer	12	
Avg. time	*16.3*	
Sample size	25	



One line Two lines

BP #31

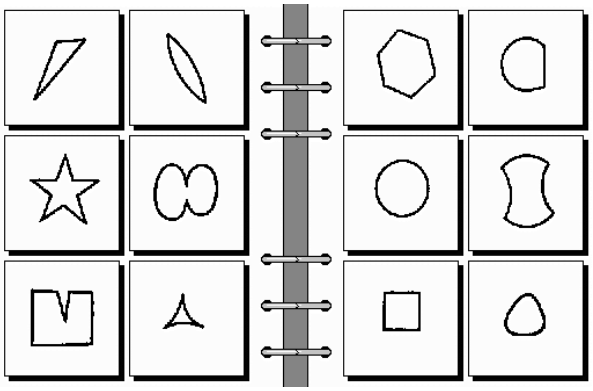
Performance:	*Human*	*Phaeaco*
Correct	18	
Avg. time	*17.9*	
Std. dev.	*9.4*	
Incorrect	1	
Avg. time	*26.0*	
No answer	5	
Avg. time	*17.6*	
Sample size	24	



Sharp projection No sharp projection

BP #32

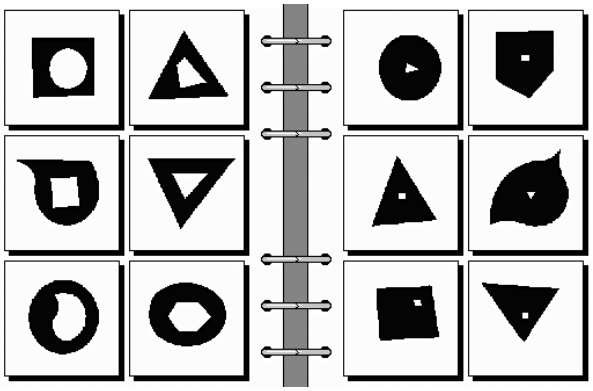
Performance:	*Human*	*Phaeaco*
Correct	6	
Avg. time	*14.7*	
Std. dev.	*6.2*	
Incorrect	1	
Avg. time	*9.0*	
No answer	17	
Avg. time	*27.2*	
Sample size	24	



Acute angle | No acute angle

BP #33

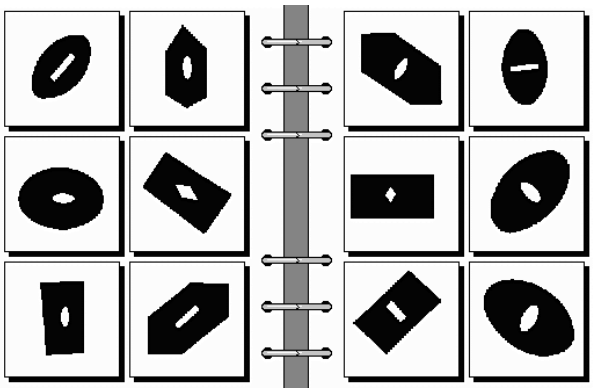
Performance:	*Human*	*Phaeaco*
Correct	5	
Avg. time	*48.6*	
Std. dev.	*28.4*	
Incorrect	3	
Avg. time	*19.7*	
No answer	22	
Avg. time	*37.4*	
Sample size	30	



Large hole | Small hole

BP #34

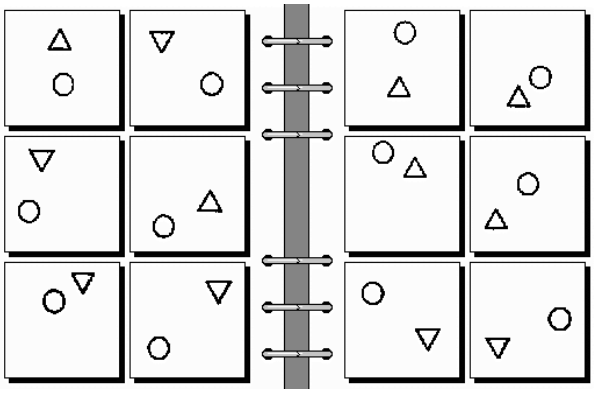
Performance:	*Human*	*Phaeaco*
Correct	30	
Avg. time	*9.1*	
Std. dev.	*5.6*	
Incorrect	1	
Avg. time	*19.0*	
No answer	0	
Avg. time		
Sample size	31	



Hole parallel to figure axis | Hole perpendicular to figure axis

BP #35

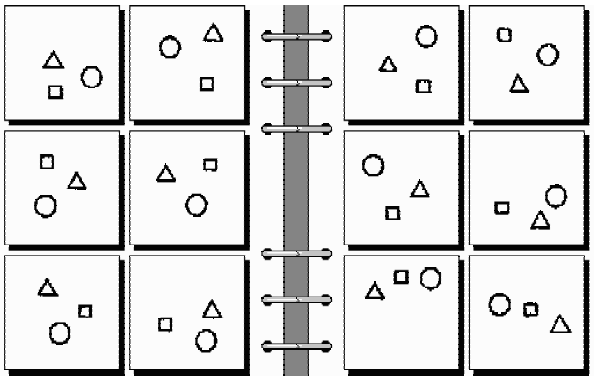
Performance:	*Human*	*Phaeaco*
Correct	11	
Avg. time	*28.2*	
Std. dev.	*17.7*	
Incorrect	1	
Avg. time	*21.0*	
No answer	14	
Avg. time	*19.0*	
Sample size	26	



Triangle above circle Circle above triangle

BP #36

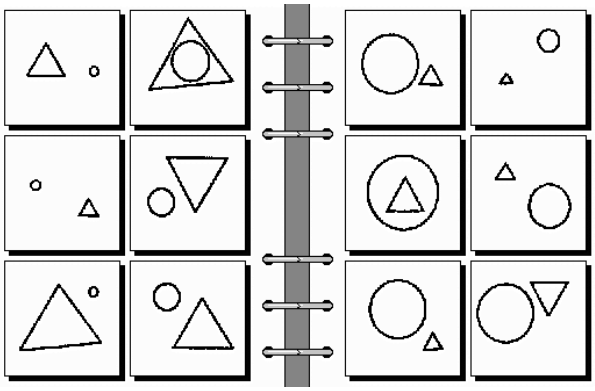
Performance:	*Human*	*Phaeaco*
Correct	23	
Avg. time	*20.1*	
Std. dev.	*14.6*	
Incorrect	2	
Avg. time	*69*	
No answer	5	
Avg. time	*23.6*	
Sample size	30	



Triangle above circle Circle above triangle

BP #37

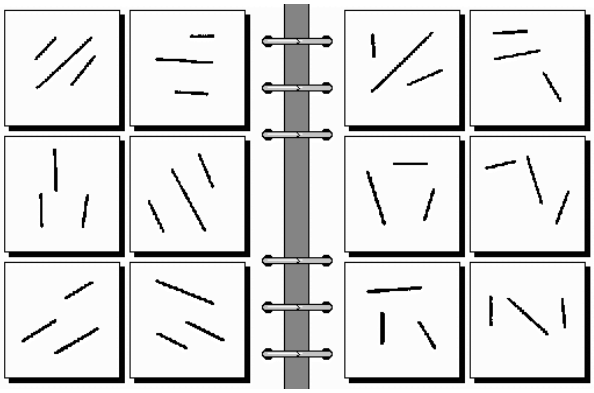
Performance:	*Human*	*Phaeaco*
Correct	3	
Avg. time	*42.0*	
Std. dev.	*24.5*	
Incorrect	5	
Avg. time	*27.2*	
No answer	17	
Avg. time	*28.5*	
Sample size	25	



Triangle larger than circle Circle larger than triangle

BP #38

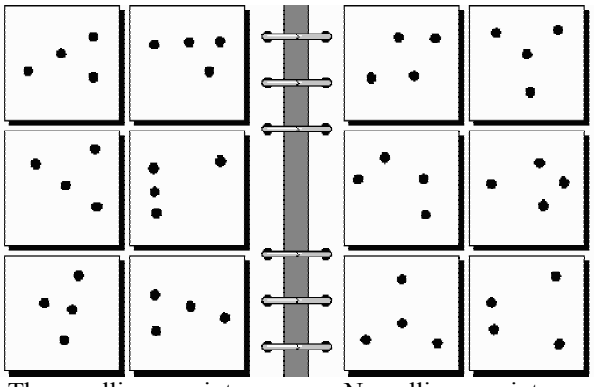
Performance:	*Human*	*Phaeaco*
Correct	24	
Avg. time	*19.8*	
Std. dev.	*11.0*	
Incorrect	2	
Avg. time	*19.0*	
No answer	4	
Avg. time	*16.3*	
Sample size	30	



All lines almost parallel Not all lines parallel

BP #39

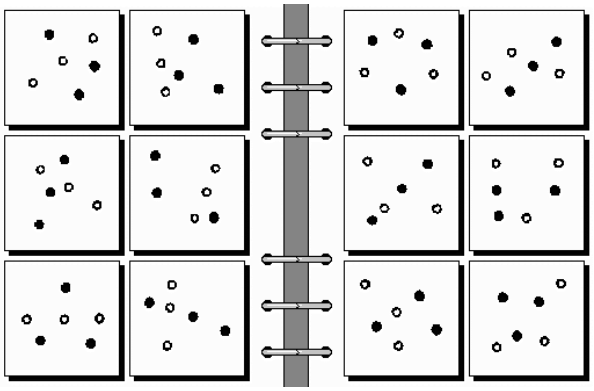
Performance:	*Human*	*Phaeaco*
Correct	30	65
Avg. time	*12.4*	*4.2*
Std. dev.	*10.0*	*0.5*
Incorrect	0	29
Avg. time		*3.1*
No answer	1	6
Avg. time	*5.0*	*5.4*
Sample size	31	100



Three collinear points No collinear points

BP #40

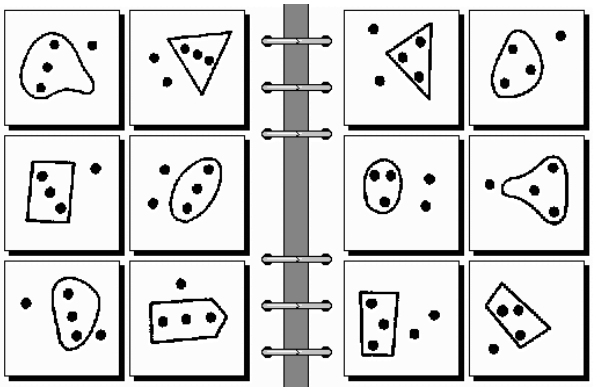
Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*29.1*	
Std. dev.	*12.1*	
Incorrect	11	
Avg. time	*25.5*	
No answer	4	
Avg. time	*29.0*	
Sample size	30	



White points are collinear

BP #41

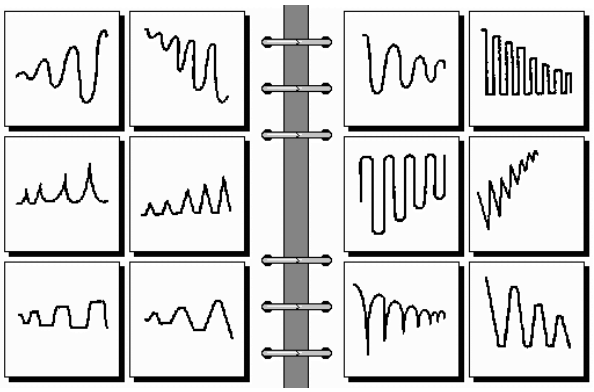
Performance:	*Human*	*Phaeaco*
Correct	14	
Avg. time	*26.0*	
Std. dev.	*11.0*	
Incorrect	1	
Avg. time	*15.0*	
No answer	11	
Avg. time	*12.7*	
Sample size	26	



Inside points collinear

BP #42

Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*13.1*	
Std. dev.	*5.8*	
Incorrect	1	
Avg. time	*44.0*	
No answer	7	
Avg. time	*17.4*	
Sample size	23	

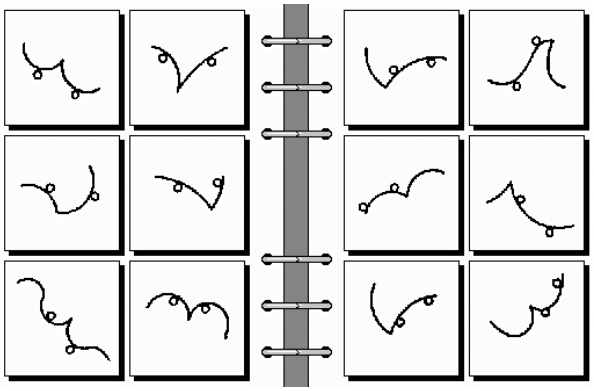


Amplitude increases from left to right

Amplitude decreases from left to right

BP #43

Performance:	*Human*	*Phaeaco*
Correct	14	
Avg. time	*14.0*	
Std. dev.	*11.1*	
Incorrect	3	
Avg. time	*16.7*	
No answer	5	
Avg. time	*19.8*	
Sample size	22	

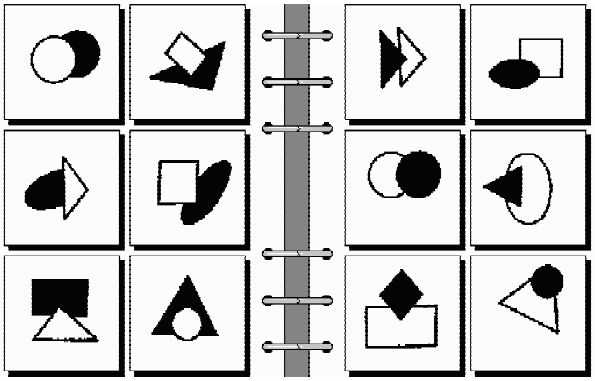


Circles on different curves

Circles on same curve

BP #44

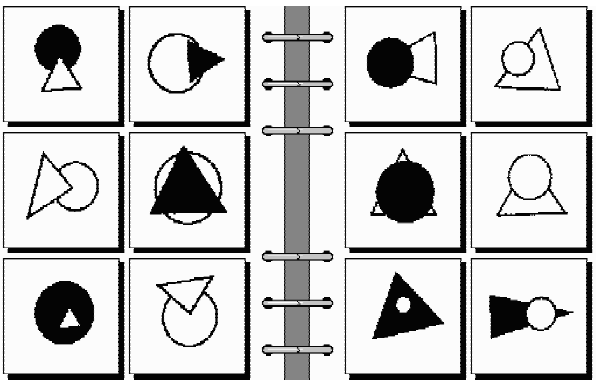
Performance:	*Human*	*Phaeaco*
Correct	6	
Avg. time	*14.0*	
Std. dev.	*10.1*	
Incorrect	2	
Avg. time	*16.5*	
No answer	13	
Avg. time	*19.9*	
Sample size	21	



Outlined on top Solid on top

BP #45

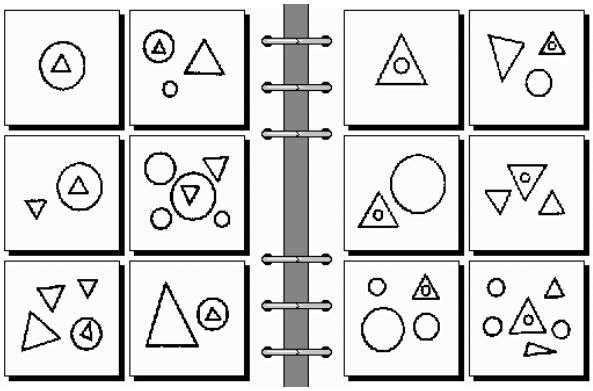
Performance:	*Human*	*Phaeaco*
Correct	29	
Avg. time	*15.0*	
Std. dev.	*12.9*	
Incorrect	0	
Avg. time		
No answer	2	
Avg. time	*42.5*	
Sample size	31	



Triangle on top Circle on top

BP #46

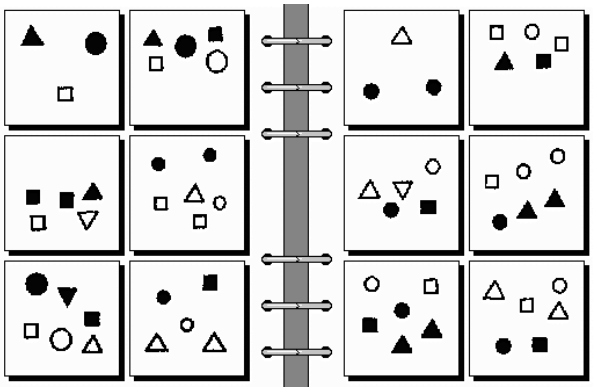
Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*17.7*	
Std. dev.	*11.0*	
Incorrect	3	
Avg. time	*45.0*	
No answer	8	
Avg. time	*16.5*	
Sample size	26	



Triangle inside circle Circle inside triangle

BP #47

Performance:	*Human*	*Phaeaco*
Correct	31	
Avg. time	*11.0*	
Std. dev.	*4.1*	
Incorrect	0	
Avg. time		
No answer	0	
Avg. time		
Sample size	31	

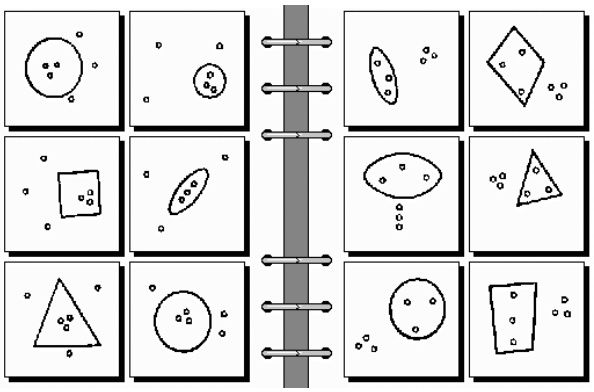


Filled objects above outlined ones

Outlined objects above filled ones

BP #48

Performance:	*Human*	*Phaeaco*
Correct	26	
Avg. time	*21.8*	
Std. dev.	*16.2*	
Incorrect	1	
Avg. time	*63*	
No answer	3	
Avg. time	*33.7*	
Sample size	30	

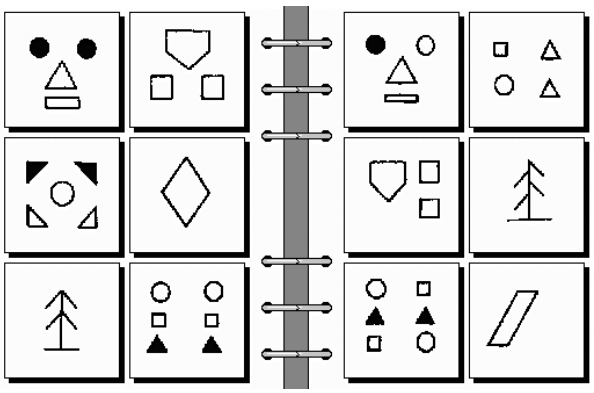


Inside points close together

Inside points far apart

BP #49

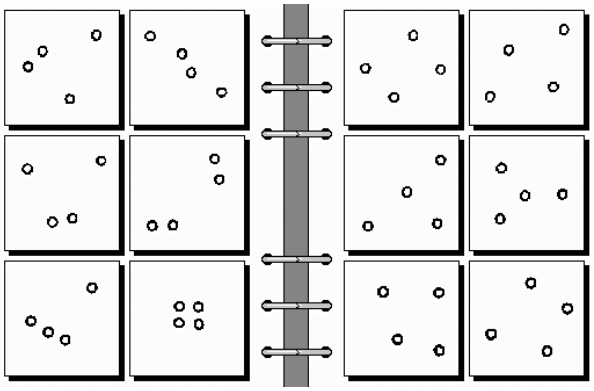
Performance:	*Human*	*Phaeaco*
Correct	23	
Avg. time	*19.3*	
Std. dev.	*8.4*	
Incorrect	0	
Avg. time		
No answer	4	
Avg. time	*7.0*	
Sample size	27	



At least one axis of symmetry

BP #50

Performance:	*Human*	*Phaeaco*
Correct	7	
Avg. time	*22.7*	
Std. dev.	*6.7*	
Incorrect	5	
Avg. time	*21.4*	
No answer	16	
Avg. time	*30.5*	
Sample size	28	

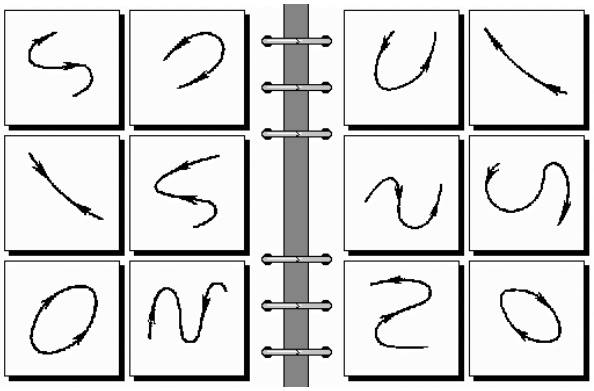


At least two circles close to each other

No two circles close to each other

BP #51

Performance:	*Human*	*Phaeaco*
Correct	19	
Avg. time	*32.1*	
Std. dev.	*21.8*	
Incorrect	2	
Avg. time	*28.5*	
No answer	9	
Avg. time	*34.3*	
Sample size	30	

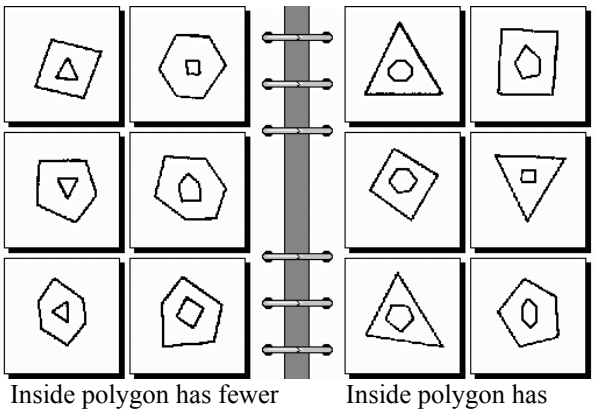


Arrows disagree

Arrows agree

BP #52

Performance:	*Human*	*Phaeaco*
Correct	16	
Avg. time	*20.6*	
Std. dev.	*16.1*	
Incorrect	0	
Avg. time		
No answer	6	
Avg. time	*15.7*	
Sample size	22	

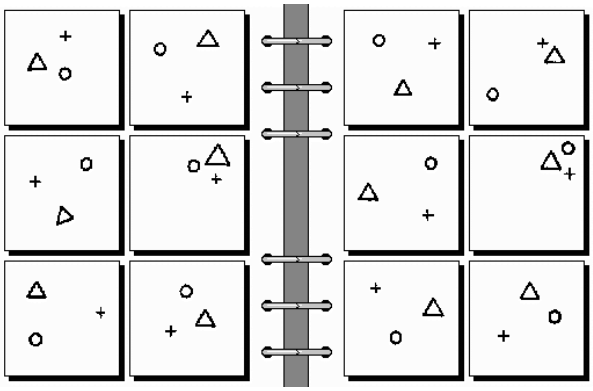


Inside polygon has fewer sides than outside

Inside polygon has more sides than outside

BP #53

Performance:	*Human*	*Phaeaco*
Correct	7	
Avg. time	*29.3*	
Std. dev.	*10.9*	
Incorrect	5	
Avg. time	*41.0*	
No answer	14	
Avg. time	*33.3*	
Sample size	26	

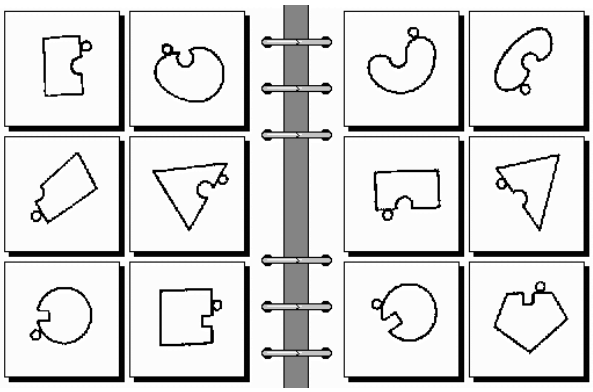


Counterclockwise: triangle, circle, cross

Clockwise: triangle, circle, cross

BP #54

Performance:	*Human*	*Phaeaco*
Correct	1	
Avg. time	*38.0*	
Std. dev.		
Incorrect	2	
Avg. time	*9.0*	
No answer	17	
Avg. time	*16.6*	
Sample size	20	

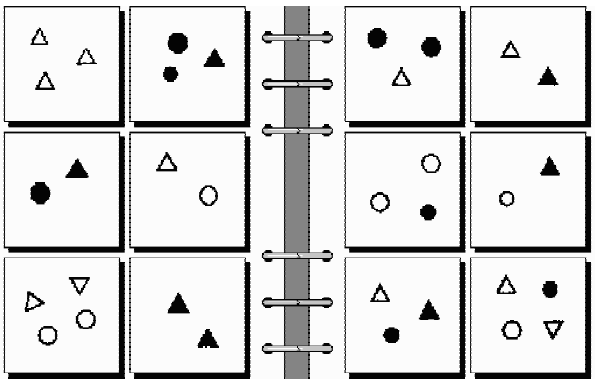


Circle left of cavity

Circle right of cavity

BP #55

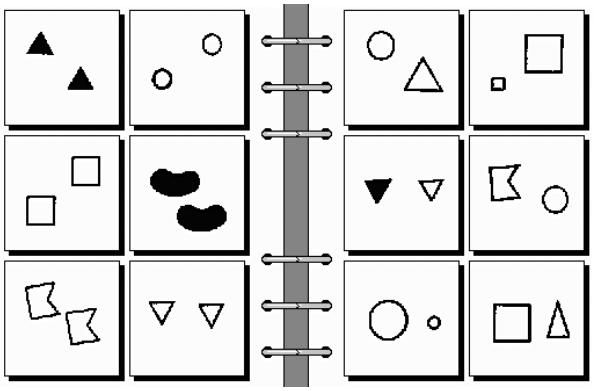
Performance:	*Human*	*Phaeaco*
Correct	6	
Avg. time	*25.0*	
Std. dev.	*10.1*	
Incorrect	2	
Avg. time	*40.0*	
No answer	10	
Avg. time	*13.4*	
Sample size	18	



All objects have the same texture

BP #56

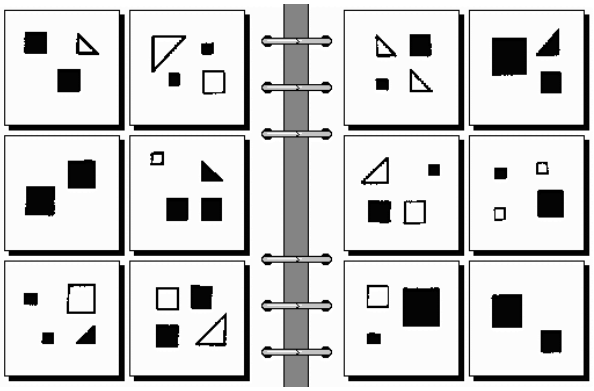
Performance:	*Human*	*Phaeaco*
Correct	22	19
Avg. time	*14.0*	*30.4*
Std. dev.	*13.1*	*7.2*
Incorrect	4	17
Avg. time	*25.3*	*20.1*
No answer	5	64
Avg. time	*25.8*	*34.9*
Sample size	31	100



Identical figures

BP #57

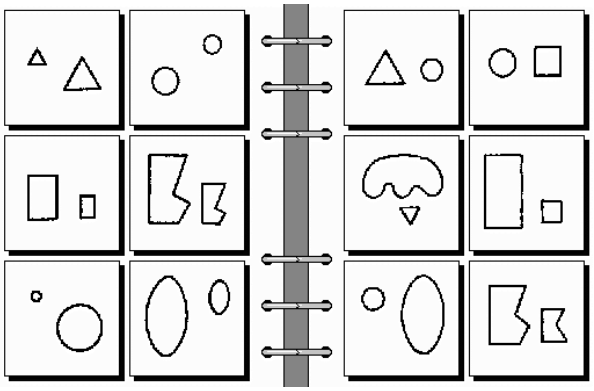
Performance:	*Human*	*Phaeaco*
Correct	14	
Avg. time	*16.1*	
Std. dev.	*9.5*	
Incorrect	13	
Avg. time	*16.2*	
No answer	2	
Avg. time	*5.5*	
Sample size	29	



Two identical filled squares

BP #58

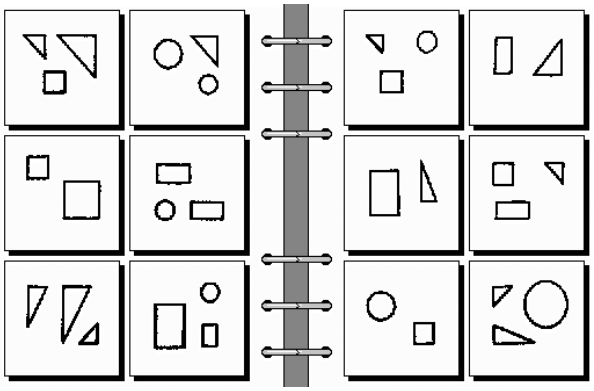
Performance:	*Human*	*Phaeaco*
Correct	4	
Avg. time	*10.0*	
Std. dev.	*1.4*	
Incorrect	2	
Avg. time	*22.0*	
No answer	10	
Avg. time	*16.6*	
Sample size	16	



Similar objects

BP #59

Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*13.3*	
Std. dev.	*6.4*	
Incorrect	0	
Avg. time		
No answer	3	
Avg. time	*7.3*	
Sample size	18	

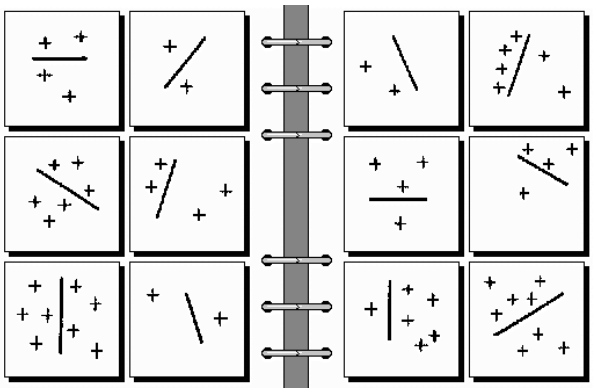


Some similar figures

No similar figures

BP #60

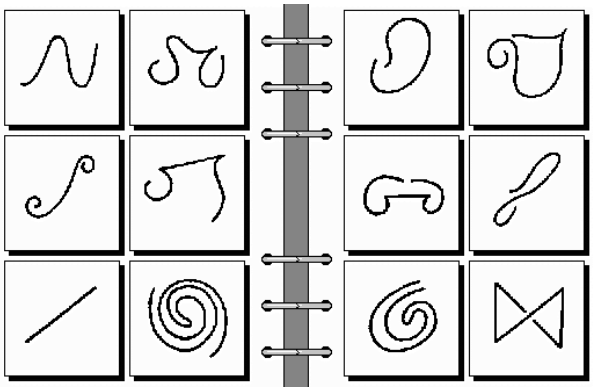
Performance:	*Human*	*Phaeaco*
Correct	5	
Avg. time	*26.0*	
Std. dev.	*15.3*	
Incorrect	2	
Avg. time	*21.0*	
No answer	9	
Avg. time	*10.9*	
Sample size	16	



Equal number of crosses on the two sides of the line

BP #61

Performance:	*Human*	*Phaeaco*
Correct	25	
Avg. time	*14.7*	
Std. dev.	*7.7*	
Incorrect	0	
Avg. time		
No answer	5	
Avg. time	*29.2*	
Sample size	30	

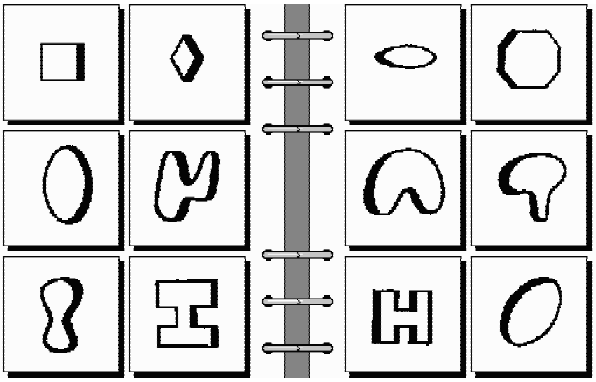


Ends of curve are far apart

Ends of curve are close together

BP #62

Performance:	*Human*	*Phaeaco*
Correct	8	
Avg. time	*34.0*	
Std. dev.	*16.0*	
Incorrect	2	
Avg. time	*15.5*	
No answer	18	
Avg. time	*29.3*	
Sample size	28	

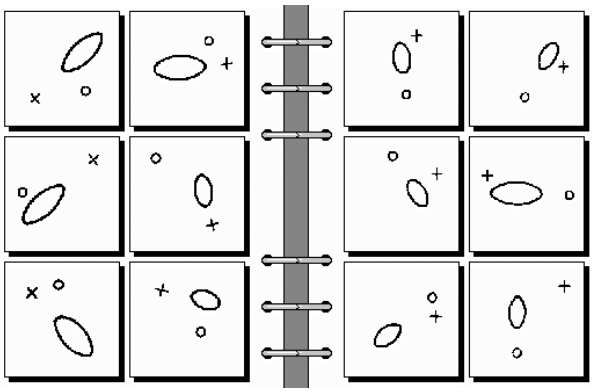


Object shaded on the right side

Object shaded on the left side

BP #63

Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*9.5*	
Std. dev.	*4.2*	
Incorrect	0	
Avg. time		
No answer	1	
Avg. time	*19.0*	
Sample size	16	

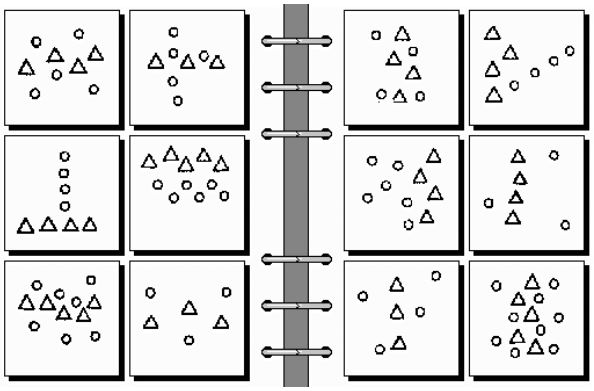


Ellipse point to the cross

Ellipse points to the circle

BP #64

Performance:	*Human*	*Phaeaco*
Correct	1	
Avg. time	*19.0*	
Std. dev.		
Incorrect	2	
Avg. time	*30.5*	
No answer	12	
Avg. time	*11.75*	
Sample size	15	

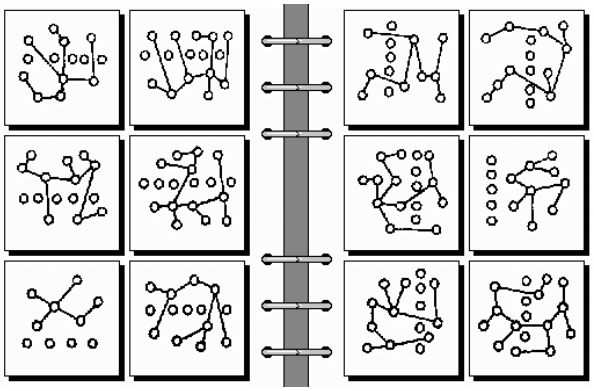


Horizontal triangle group

Vertical triangle group

BP #65

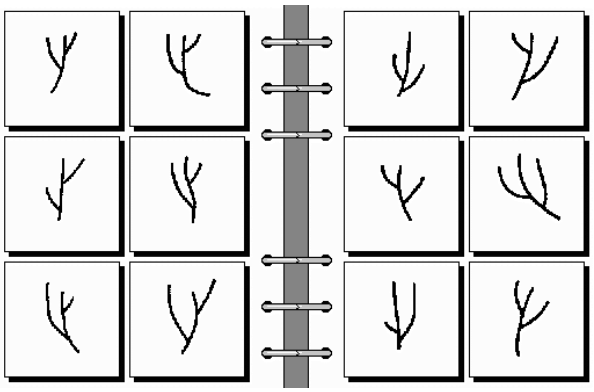
Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*20.5*	
Std. dev.	*8*	
Incorrect	2	
Avg. time	*72.0*	
No answer	13	
Avg. time	*43.3*	
Sample size	30	



Loose circles horizontal | Loose circles vertical

BP #66

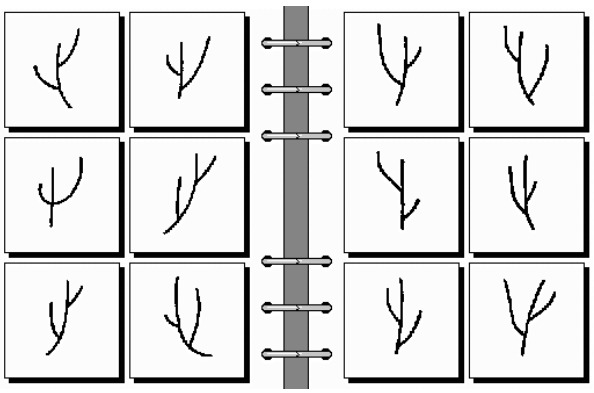
Performance:	*Human*	*Phaeaco*
Correct	21	
Avg. time	*17.5*	
Std. dev.	*10.1*	
Incorrect	0	
Avg. time		
No answer	8	
Avg. time	*13.8*	
Sample size	29	



Right branch begins higher than left branch | Right branch begins lower than left branch

BP #67

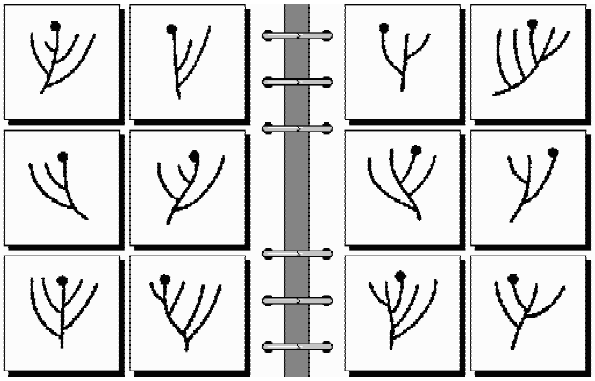
Performance:	*Human*	*Phaeaco*
Correct	9	
Avg. time	*34.3*	
Std. dev.	*16.3*	
Incorrect	7	
Avg. time	*37.9*	
No answer	13	
Avg. time	*21.3*	
Sample size	29	



Right branch ends higher than left branch | Right branch ends lower than left branch

BP #68

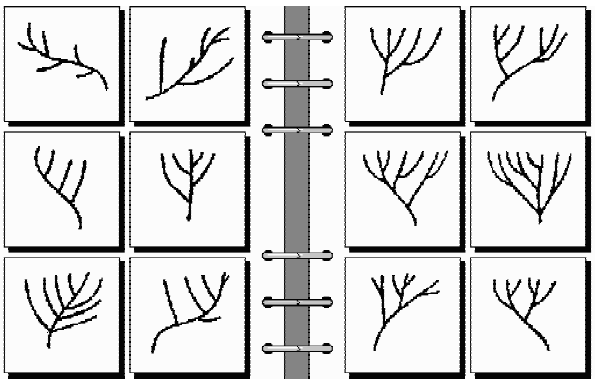
Performance:	*Human*	*Phaeaco*
Correct	0	
Avg. time		
Std. dev.		
Incorrect	3	
Avg. time	*17.0*	
No answer	11	
Avg. time	*15.7*	
Sample size	14	



Dot on tip of trunk | Dot on tip of branch

BP #69

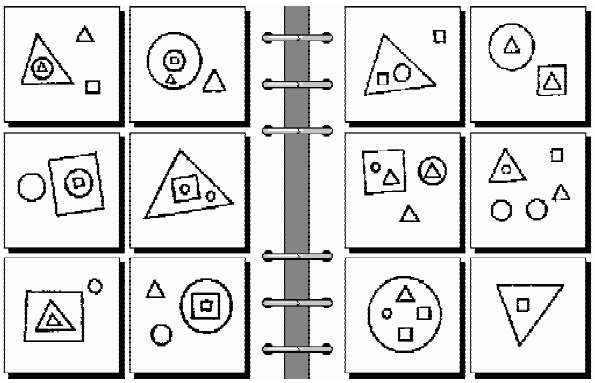
Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*10.5*	
Std. dev.	*5.3*	
Incorrect	0	
Avg. time		
No answer	1	
Avg. time	*18.0*	
Sample size	16	



One level of branching | Two levels of branching

BP #70

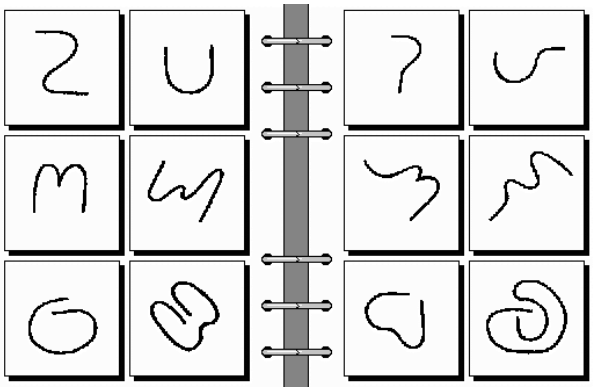
Performance:	*Human*	*Phaeaco*
Correct	19	
Avg. time	*28.4*	
Std. dev.	*17.9*	
Incorrect	3	
Avg. time	*21.3*	
No answer	7	
Avg. time	*29.9*	
Sample size	29	



Two levels of inside-ness | One level of inside-ness

BP #71

Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*34.6*	
Std. dev.	*14.3*	
Incorrect	6	
Avg. time	*42.2*	
No answer	9	
Avg. time	*41.8*	
Sample size	30	

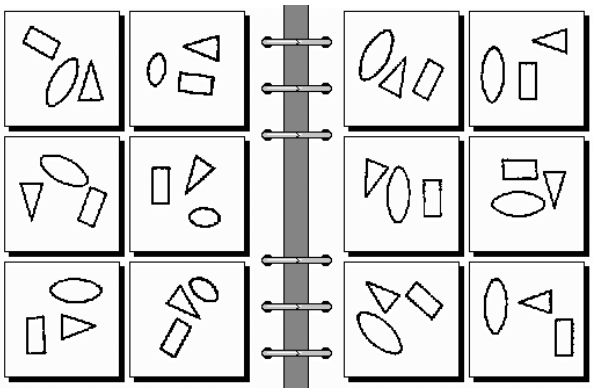


Curve ends are parallel

Curve ends are perpendicular

BP #72

Performance:	*Human*	*Phaeaco*
Correct	4	
Avg. time	*39.5*	
Std. dev.	*19.8*	
Incorrect	3	
Avg. time	*57.0*	
No answer	23	
Avg. time	*30.2*	
Sample size	30	

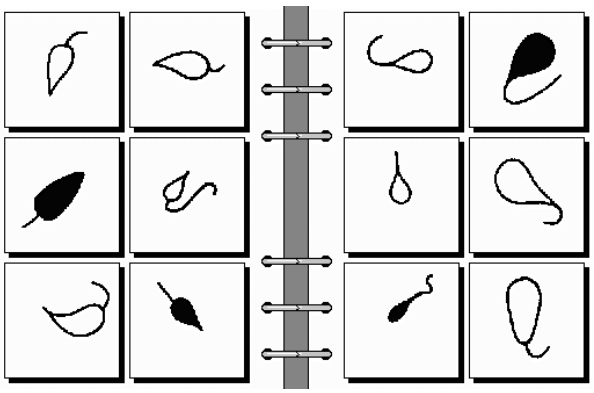


Rectangle perpendicular to ellipse

Rectangle parallel to ellipse

BP #73

Performance:	*Human*	*Phaeaco*
Correct	0	
Avg. time		
Std. dev.		
Incorrect	1	
Avg. time	*8.0*	
No answer	14	
Avg. time	*15.2*	
Sample size	15	

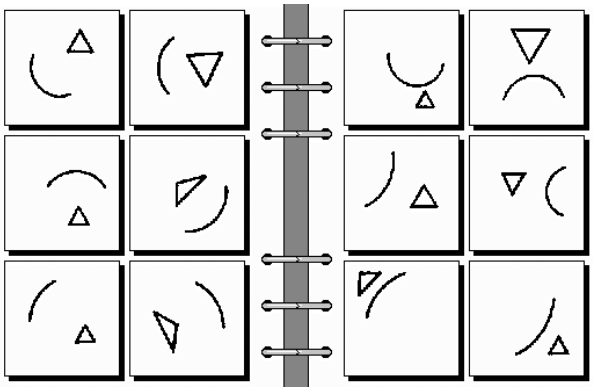


Tail at the rounded end

Tail at the pointy end

BP #74

Performance:	*Human*	*Phaeaco*
Correct	11	
Avg. time	*12.2*	
Std. dev.	*5.0*	
Incorrect	3	
Avg. time	*10.7*	
No answer	1	
Avg. time	*2.0*	
Sample size	15	

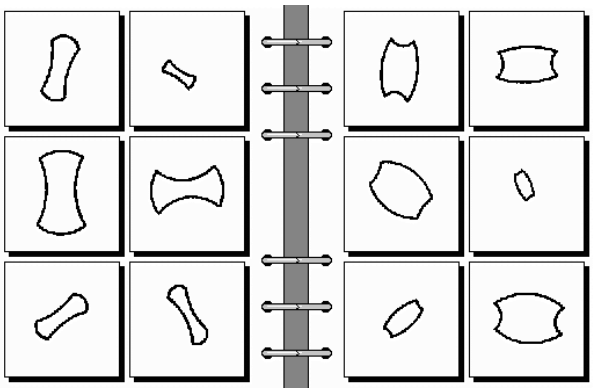


Triangle on the concave side of the arc

Triangle on the convex side of the arc

BP #75

Performance:	*Human*	*Phaeaco*
Correct	26	
Avg. time	*16.6*	
Std. dev.	*9.2*	
Incorrect	0	
Avg. time		
No answer	4	
Avg. time	*32.25*	
Sample size	30	

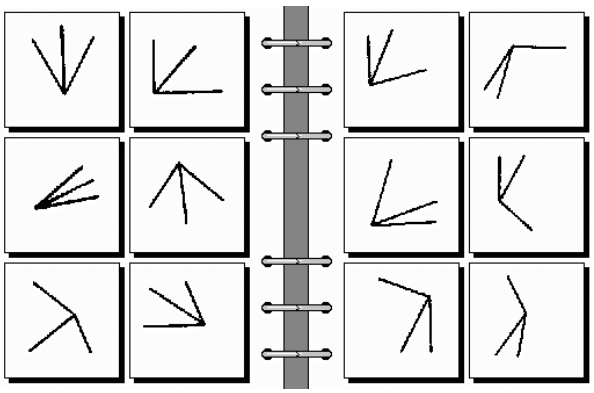


Long sides concave

Long sides convex

BP #76

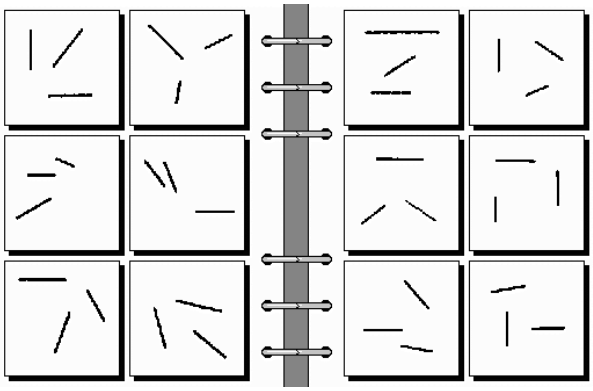
Performance:	*Human*	*Phaeaco*
Correct	9	
Avg. time	*11.4*	
Std. dev.	*5.2*	
Incorrect	1	
Avg. time	*17.0*	
No answer	4	
Avg. time	*7.3*	
Sample size	14	



Two equal angles

BP #77

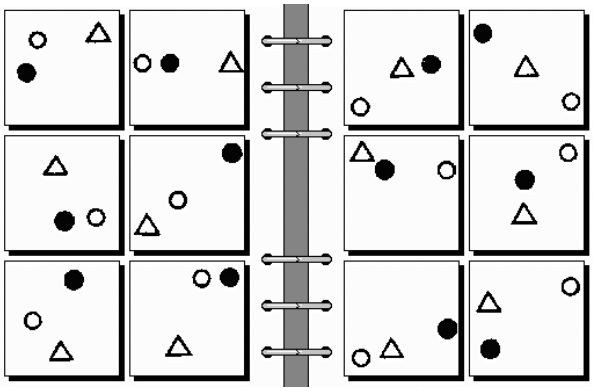
Performance:	*Human*	*Phaeaco*
Correct	18	
Avg. time	*29.6*	
Std. dev.	*22.0*	
Incorrect	3	
Avg. time	*24.3*	
No answer	9	
Avg. time	*31.2*	
Sample size	30	



Lines meet at imaginary point

BP #78

Performance:	*Human*	*Phaeaco*
Correct	8	
Avg. time	*32.4*	
Std. dev.	*14.5*	
Incorrect	0	
Avg. time		
No answer	22	
Avg. time	*29.0*	
Sample size	30	

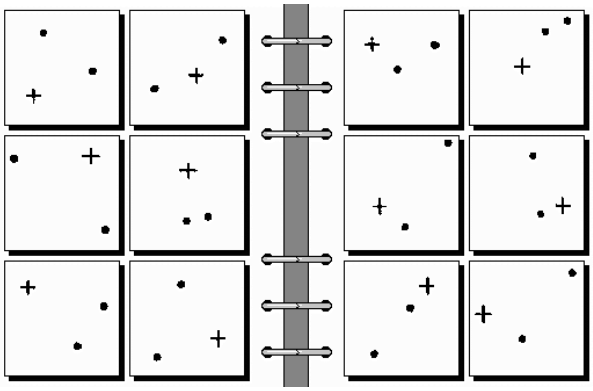


Filled circle closer to outlined circle

Filled circle closer to triangle

BP #79

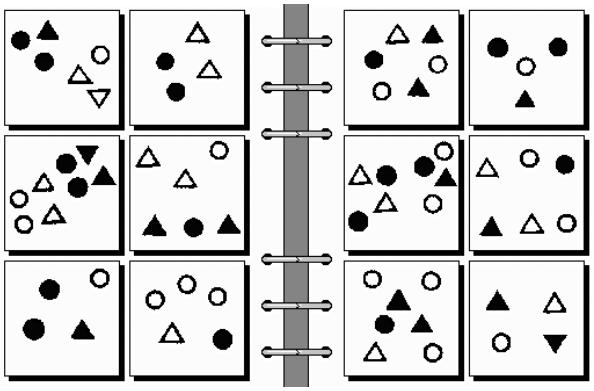
Performance:	*Human*	*Phaeaco*
Correct	1	
Avg. time	*26.0*	
Std. dev.		
Incorrect	0	
Avg. time		
No answer	15	
Avg. time	*11.6*	
Sample size	16	



Dots equidistant from cross

BP #80

Performance:	*Human*	*Phaeaco*
Correct	1	
Avg. time	*26.0*	
Std. dev.		
Incorrect	0	
Avg. time		
No answer	14	
Avg. time	*11.5*	
Sample size	15	

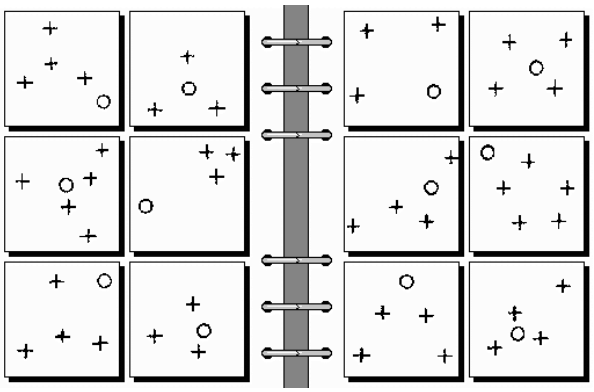


Filled and outlined groups are separate

Filled and outlined groups overlap

BP #81

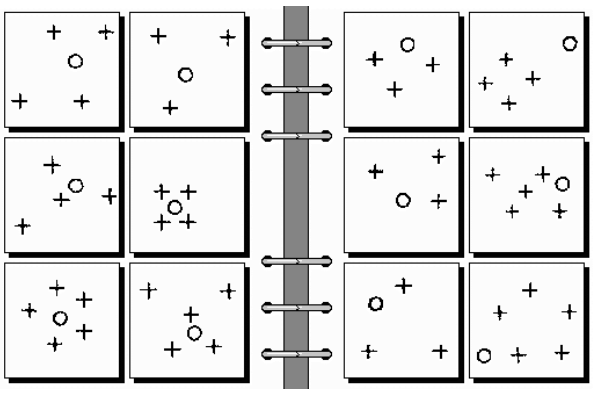
Performance:	*Human*	*Phaeaco*
Correct	13	
Avg. time	*13.7*	
Std. dev.	*6.2*	
Incorrect	6	
Avg. time	*29.0*	
No answer	12	
Avg. time	*55.0*	
Sample size	31	



Convex hull of crosses is equilateral triangle

BP #82

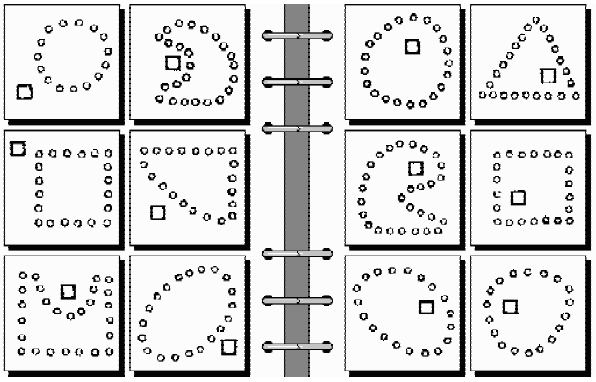
Performance:	*Human*	*Phaeaco*
Correct	0	
Avg. time		
Std. dev.		
Incorrect	1	
Avg. time	*66.0*	
No answer	29	
Avg. time	*34.9*	
Sample size	30	



Circle in convex hull of crosses

BP #83

Performance:	*Human*	*Phaeaco*
Correct	22	
Avg. time	*23.6*	
Std. dev.	*11.4*	
Incorrect	2	
Avg. time	*16.0*	
No answer	6	
Avg. time	*19.5*	
Sample size	30	

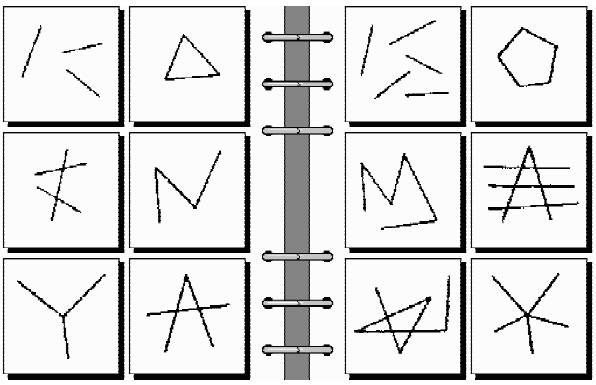


Square out of region made by circles

Square in region made by circles

BP #84

Performance:	*Human*	*Phaeaco*
Correct	31	
Avg. time	*12.6*	
Std. dev.	*7.6*	
Incorrect	0	
Avg. time		
No answer	0	
Avg. time		
Sample size	31	

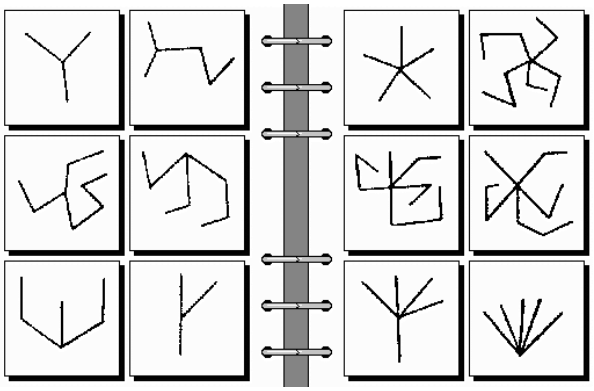


Three lines

Five lines

BP #85

Performance:	*Human*	*Phaeaco*
Correct	27	94
Avg. time	*20.5*	*11.5*
Std. dev.	*8.1*	*4.3*
Incorrect	2	0
Avg. time	*34.5*	
No answer	1	6
Avg. time	*2.0*	*13.1*
Sample size	30	100

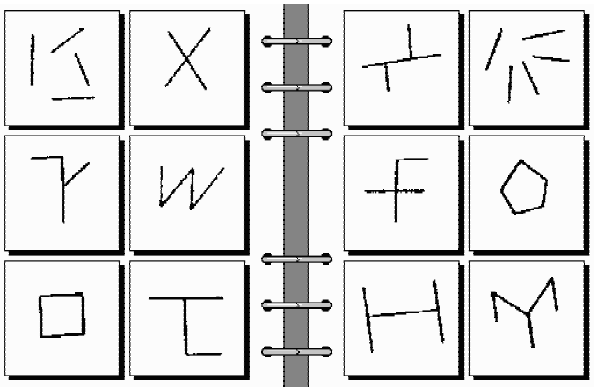


Three line strings meeting at a K-point

Five line strings meeting at a K-point

BP #86

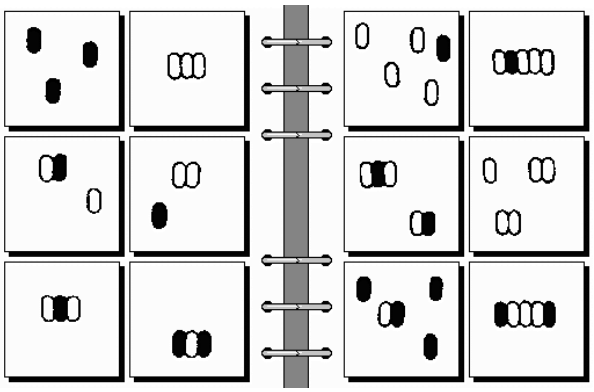
Performance:	*Human*	*Phaeaco*
Correct	8	
Avg. time	*45.3*	
Std. dev.	*25.6*	
Incorrect	8	
Avg. time	*43.6*	
No answer	15	
Avg. time	*26.7*	
Sample size	31	



Four line segments Five line segments

BP #87

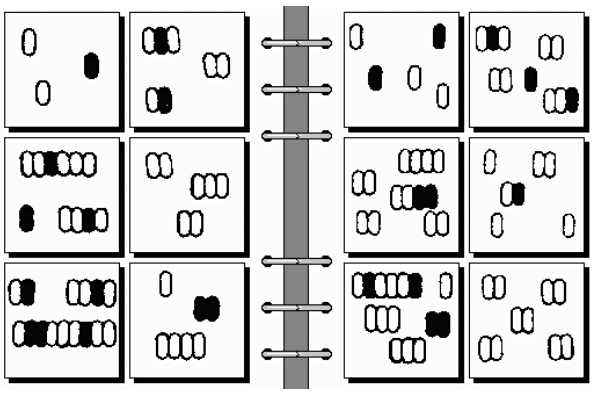
Performance:	*Human*	*Phaeaco*
Correct	16	
Avg. time	*33.4*	
Std. dev.	*16.9*	
Incorrect	2	
Avg. time	*40.0*	
No answer	12	
Avg. time	*29.8*	
Sample size	30	



Three ovals Five ovals

BP #88

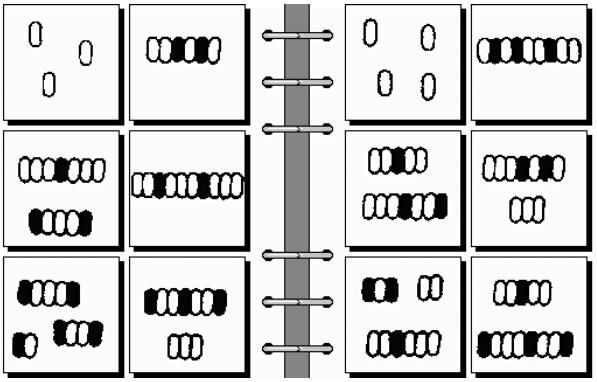
Performance:	*Human*	*Phaeaco*
Correct	16	
Avg. time	*21.6*	
Std. dev.	*16.0*	
Incorrect	0	
Avg. time		
No answer	14	
Avg. time	*17.6*	
Sample size	30	



Three groups of ovals Five groups of ovals

BP #89

Performance:	*Human*	*Phaeaco*
Correct	6	
Avg. time	*35.7*	
Std. dev.	*24.2*	
Incorrect	5	
Avg. time	*62.0*	
No answer	19	
Avg. time	*30.8*	
Sample size	30	

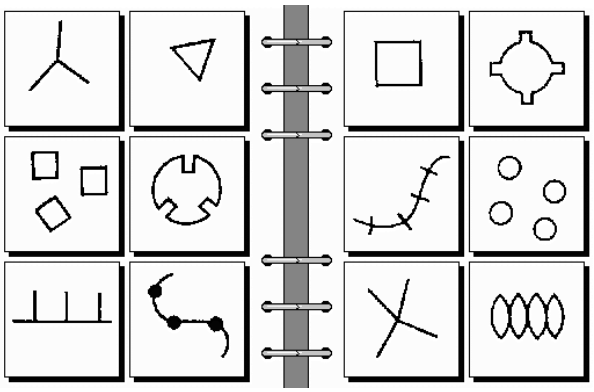


Three groups of outlined ovals

Four groups of outlined ovals

BP #90

Performance:	*Human*	*Phaeaco*
Correct	2	
Avg. time	*36.0*	
Std. dev.	*19.8*	
Incorrect	0	
Avg. time		
No answer	12	
Avg. time	*17.7*	
Sample size	14	

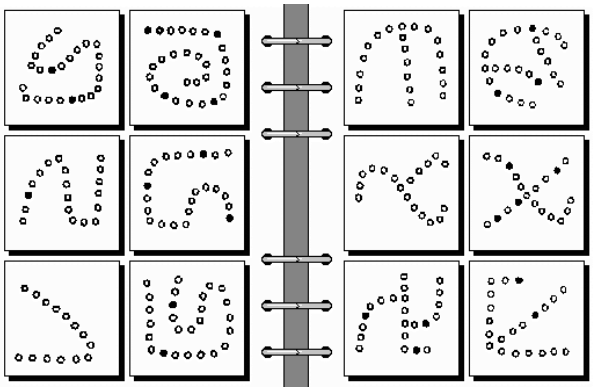


Three

Four

BP #91

Performance:	*Human*	*Phaeaco*
Correct	18	
Avg. time	*32.0*	
Std. dev.	*15.4*	
Incorrect	0	
Avg. time		
No answer	12	
Avg. time	*22.8*	
Sample size	30	

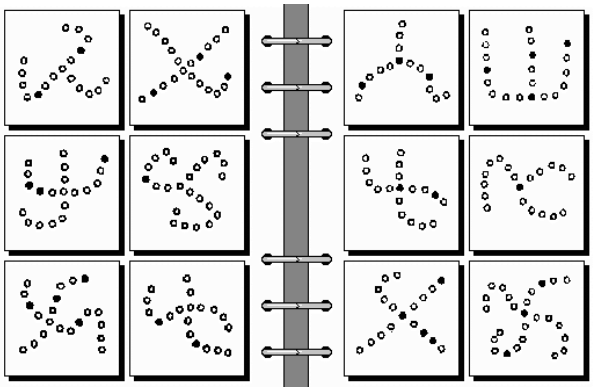


One curve

More than one curve

BP #92

Performance:	*Human*	*Phaeaco*
Correct	6	
Avg. time	*21.0*	
Std. dev.	*12.6*	
Incorrect	0	
Avg. time		
No answer	9	
Avg. time	*11.1*	
Sample size	15	

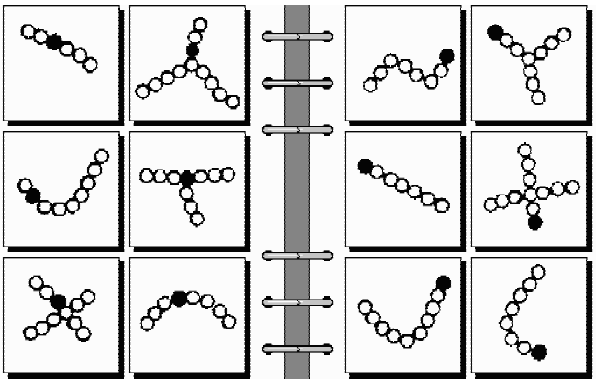


Outlined circle at cross point

Filled circle at cross point

BP #93

Performance:	*Human*	*Phaeaco*
Correct	3	
Avg. time	*27.7*	
Std. dev.	*2.5*	
Incorrect	0	
Avg. time		
No answer	17	
Avg. time	*15.4*	
Sample size	20	

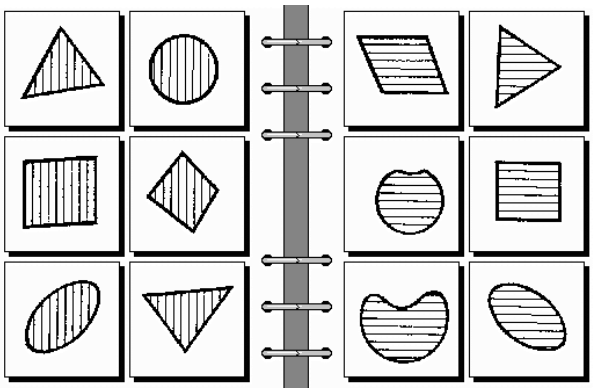


Filled circle not at endpoint

Filled circle at endpoint

BP #94

Performance:	*Human*	*Phaeaco*
Correct	15	
Avg. time	*7.5*	
Std. dev.	*3.8*	
Incorrect	0	
Avg. time		
No answer	0	
Avg. time		
Sample size	15	

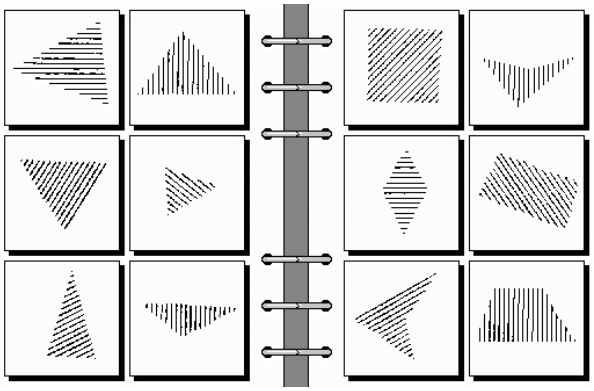


Texture made of vertical lines

Texture made of horizontal lines

BP #95

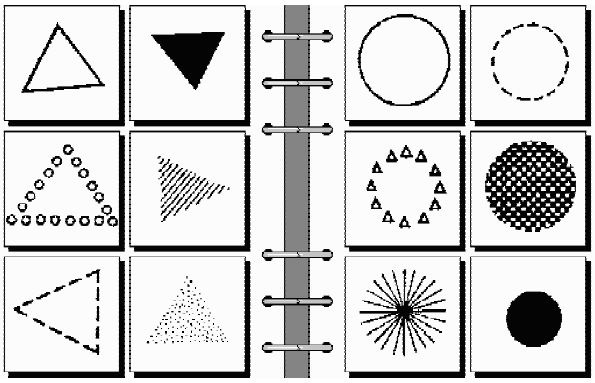
Performance:	*Human*	*Phaeaco*
Correct	30	
Avg. time	*8.4*	
Std. dev.	*5.1*	
Incorrect	0	
Avg. time		
No answer	1	
Avg. time	*17.0*	
Sample size	31	



Triangle Quadrilateral

BP #96

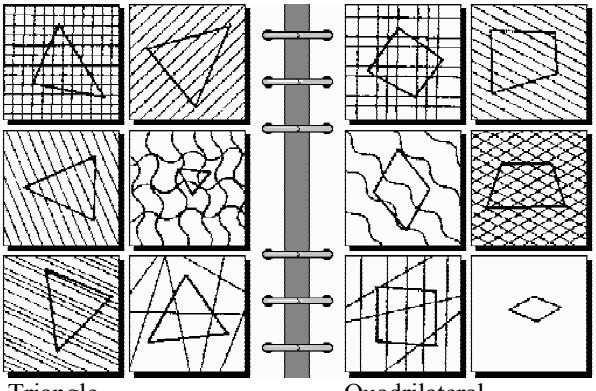
Performance:	*Human*	*Phaeaco*
Correct	25	
Avg. time	*16.9*	
Std. dev.	*13.3*	
Incorrect	0	
Avg. time		
No answer	5	
Avg. time	*16.0*	
Sample size	30	



Triangle Circle

BP #97

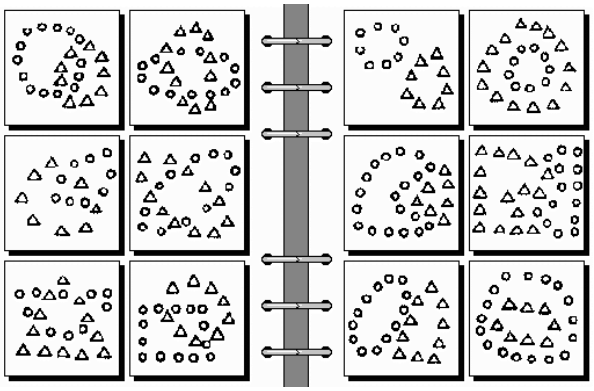
Performance:	*Human*	*Phaeaco*
Correct	29	
Avg. time	*8.7*	
Std. dev.	*5.6*	
Incorrect	0	
Avg. time		
No answer	2	
Avg. time	*18.0*	
Sample size	31	



Triangle Quadrilateral

BP #98

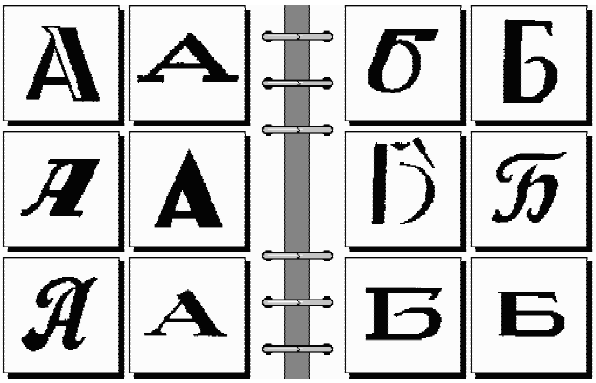
Performance:	*Human*	*Phaeaco*
Correct	9	
Avg. time	*12.1*	
Std. dev.	*8.7*	
Incorrect	3	
Avg. time	*13.7*	
No answer	4	
Avg. time	*19.3*	
Sample size	16	



Curves made of circles and triangles intersect

BP #99

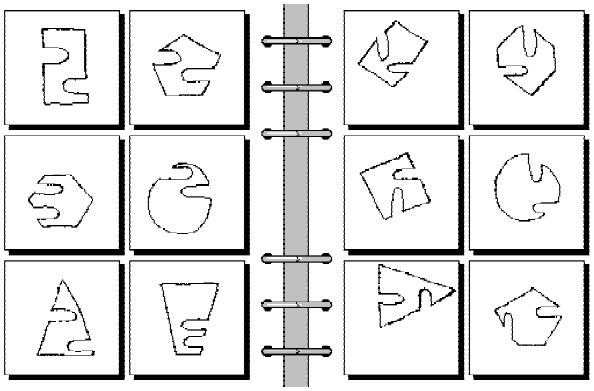
Performance:	*Human*	*Phaeaco*
Correct	9	
Avg. time	*19.8*	
Std. dev.	*9.5*	
Incorrect	1	
Avg. time	*19.0*	
No answer	6	
Avg. time	*8.3*	
Sample size	16	



Letter “A” Letter “Б”

BP #100

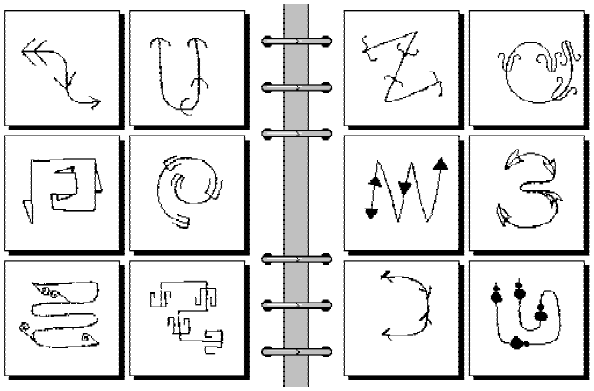
Performance:	*Human*	*Phaeaco*
Correct	14	
Avg. time	*5.4*	
Std. dev.	*3.3*	
Incorrect	0	
Avg. time		
No answer	0	
Avg. time		
Sample size	14	



Parallel dents Perpendicular dents

BP #101

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

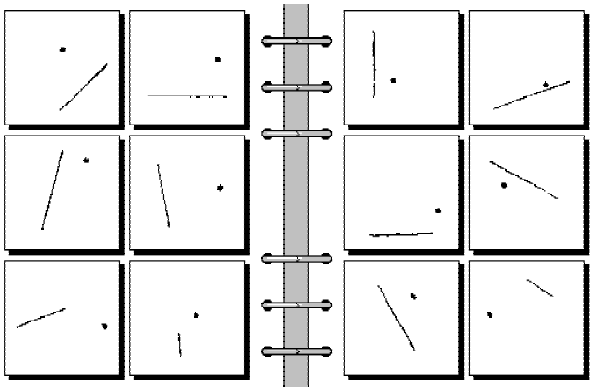


Internal arrows point outward

Internal arrows point inward

BP #102

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

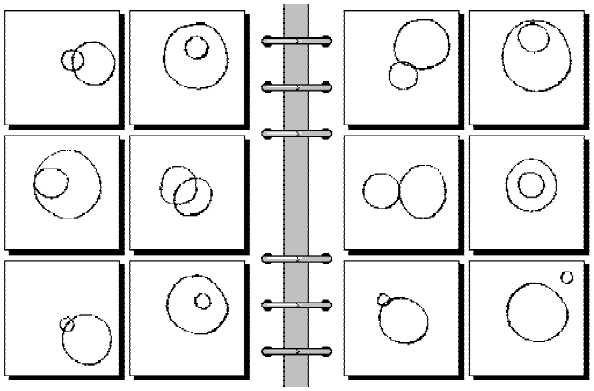


Isosceles triangle

Scalene triangle

BP #103

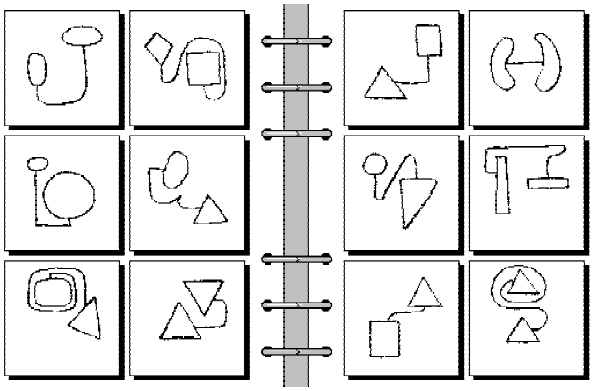
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



One circle passes through the center of the other circle

BP #104

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

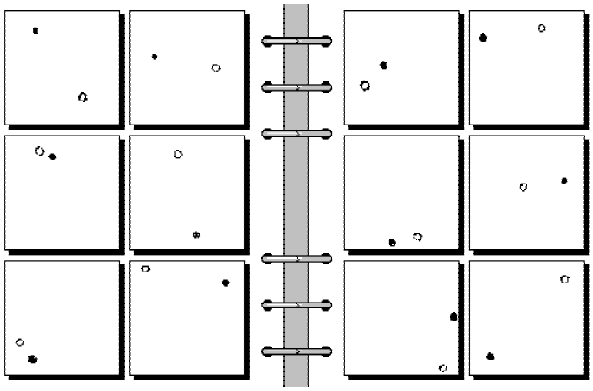


Ends of line point to same direction

Ends of line point to opposite directions

BP #105

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

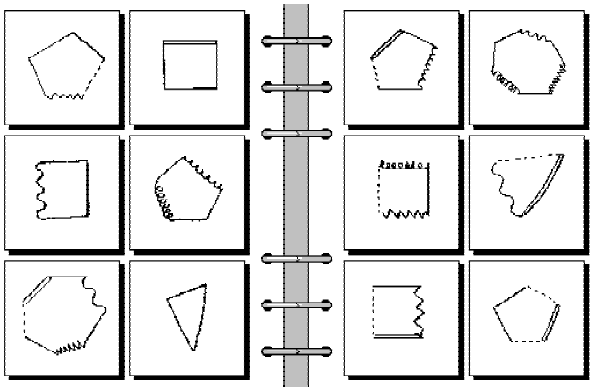


Negative slope

Positive slope

BP #106

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

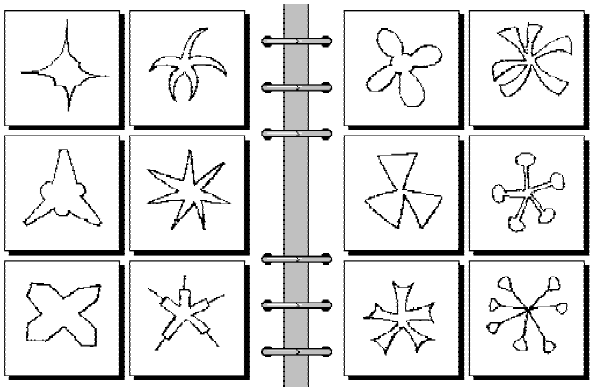


Three simple lines

Three non-simple lines

BP #107

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

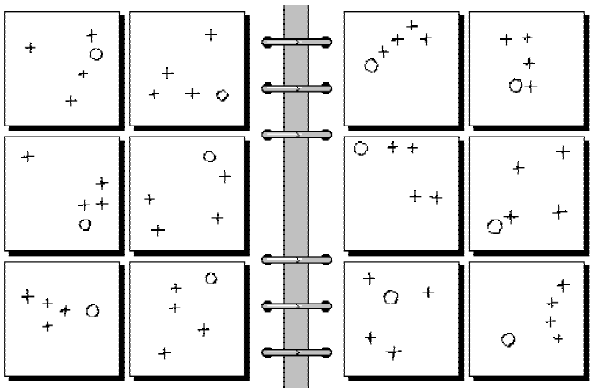


Petals taper off

Petals thicken

BP #108

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

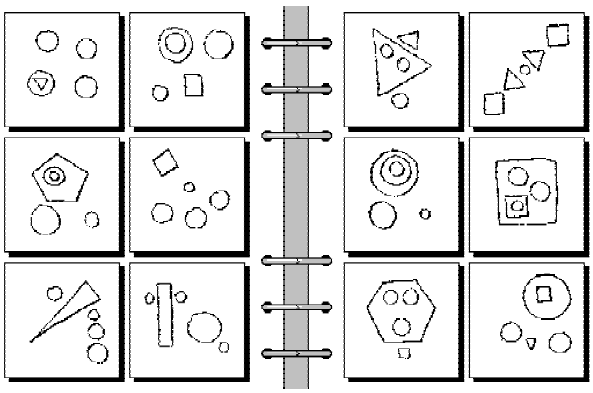


Circle on the right of the box

Circle on the left of the box

BP #109

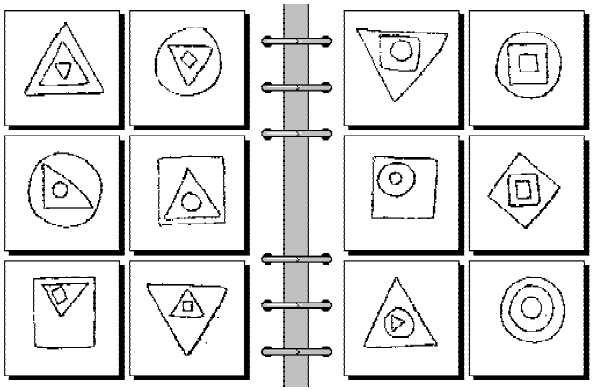
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Four circles

BP #110

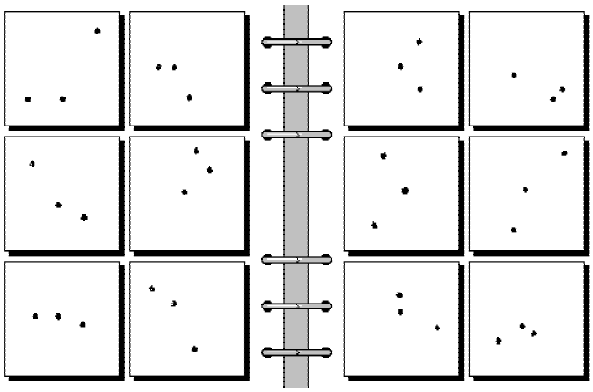
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Middle shape is triangle

BP #111

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

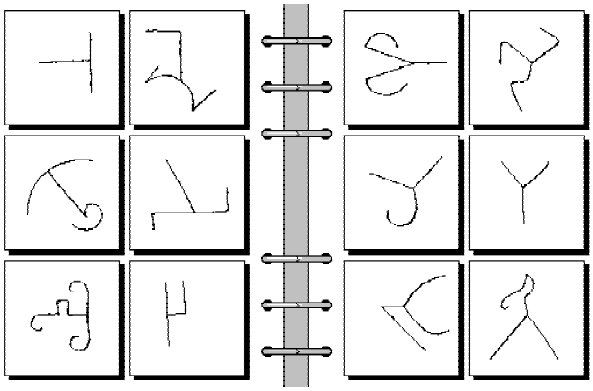


Dots equidistant along *x*-axis

Dots equidistant along *y*-axis

BP #112

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

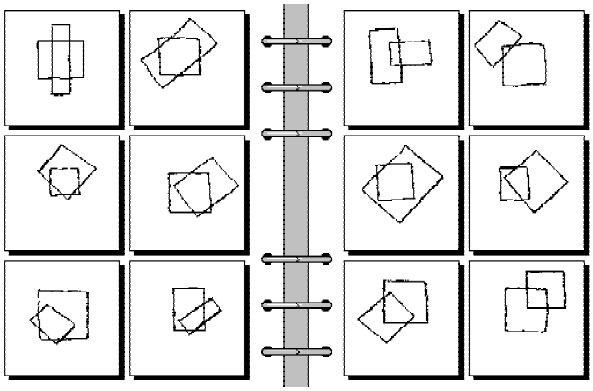


T-like point

Y-like point

BP #113

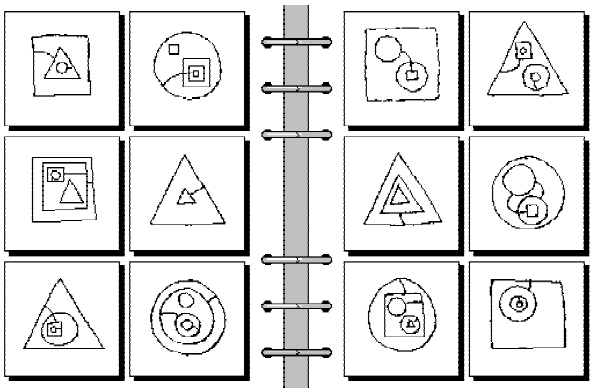
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Four X-points Two X-points

BP #114

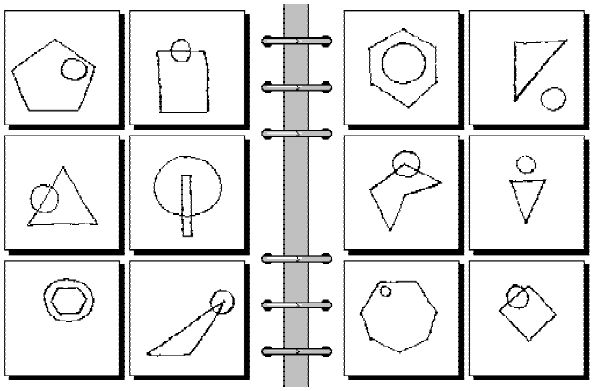
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Most deeply nested shape is reachable from outside

BP #115

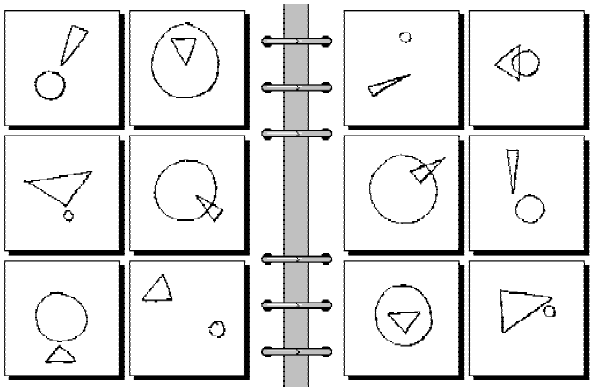
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Polygon stands on side Polygon stands on vertex

BP #116

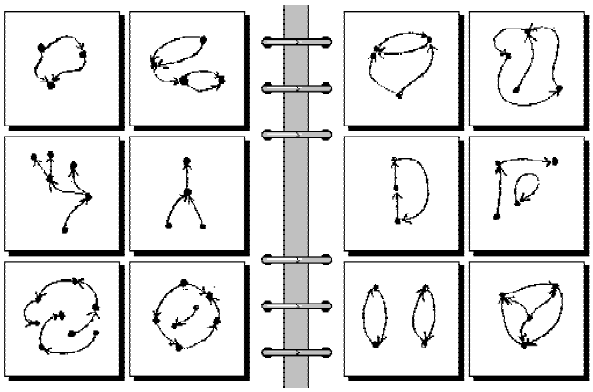
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Triangle points to center of circle

BP #117

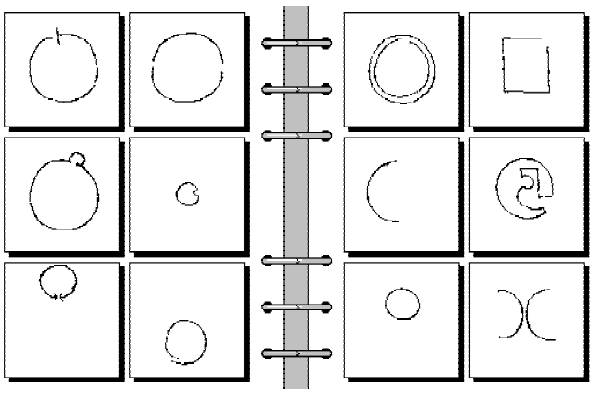
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



No cycle There is a cycle

BP #118

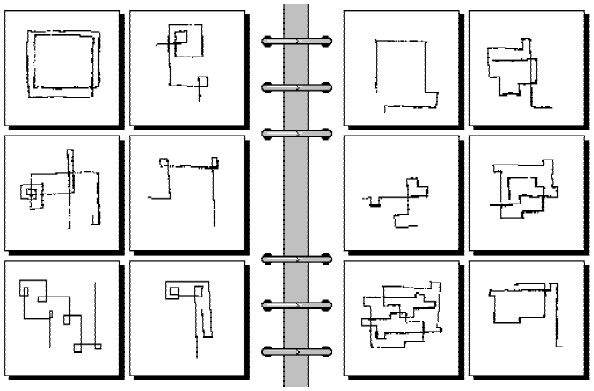
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Almost a circle

BP #119

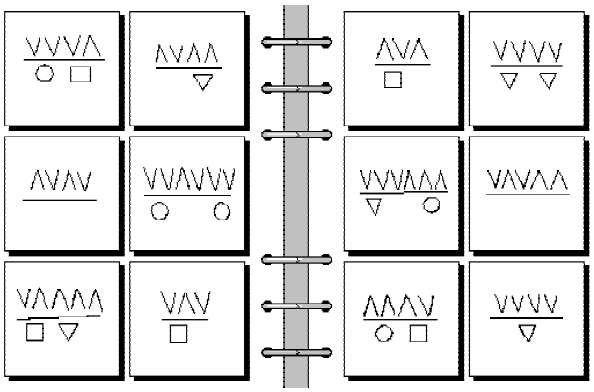
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



All turns are in one direction

BP #120

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

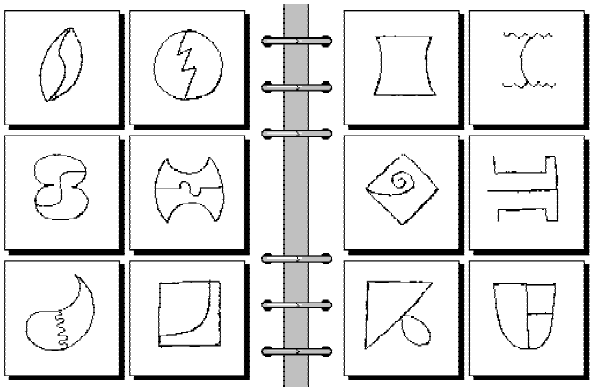


Circle: ⋁⋁; square: ⋁⋀;
triangle: ⋀⋀; blank: ⋀⋁

Circle: ⋀⋀; square: ⋀⋁;
triangle: ⋁⋁; blank: ⋁⋀

BP #121

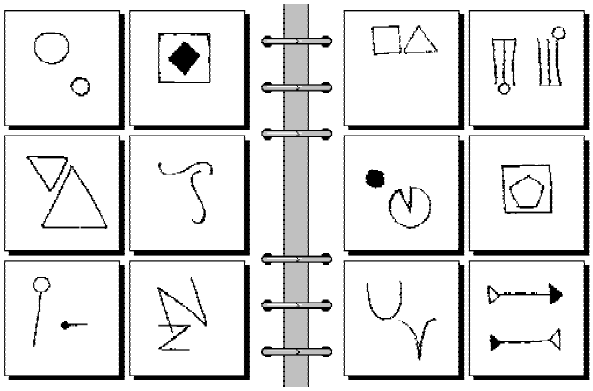
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Line joins two different
points of closed region

BP #122

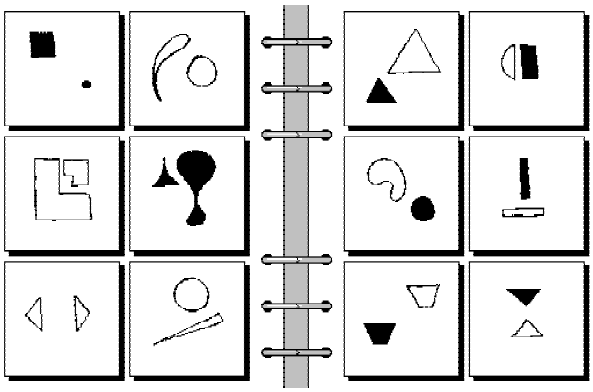
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Similar shapes Dissimilar shapes

BP #123

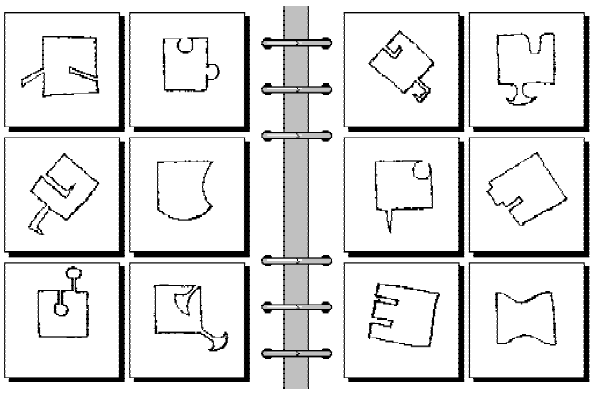
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Similar textures Dissimilar textures

BP #124

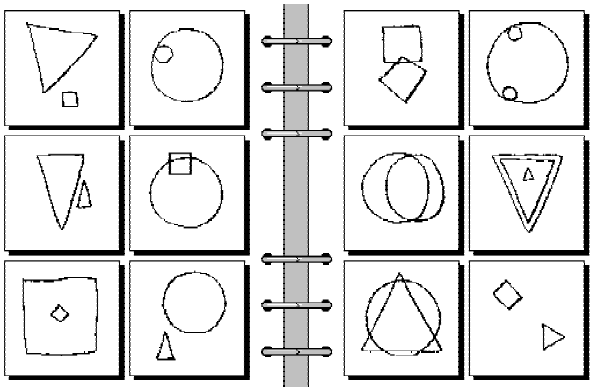
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



One protrusion and one indentation of the same shape

BP #125

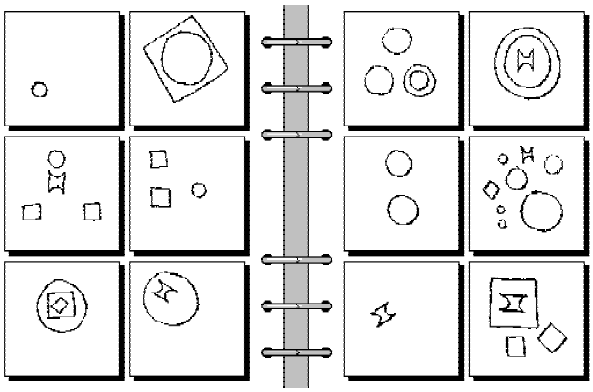
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



One large and one small object

BP #126

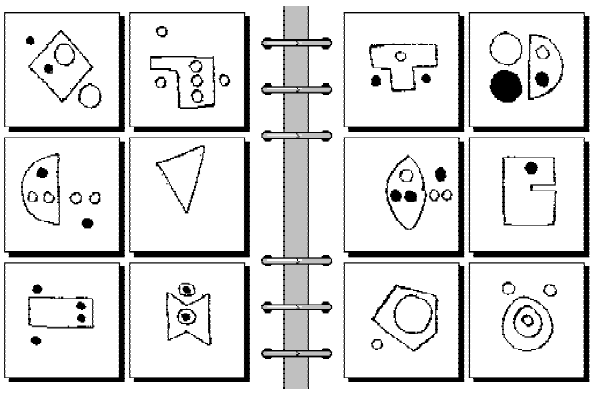
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Exactly one circle

BP #127

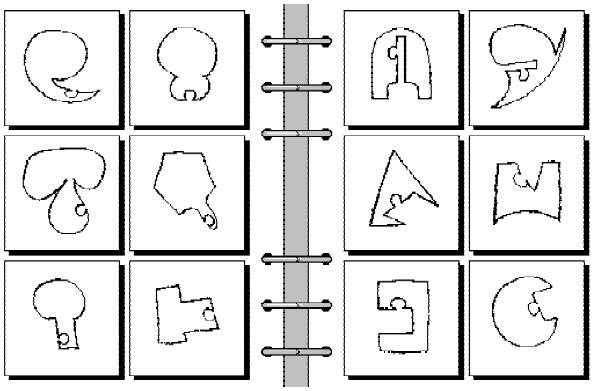
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Same objects inside and outside larger shape

BP #128

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

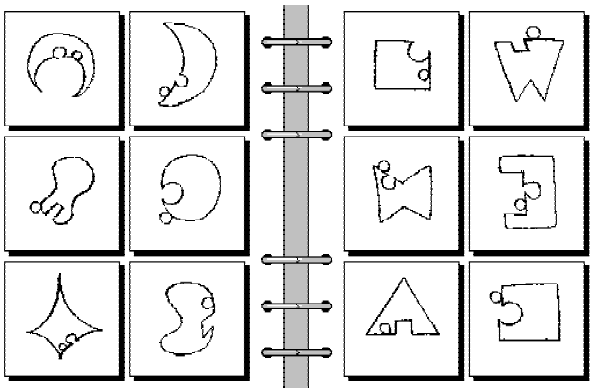


Indentation on protrusion

Indentation on indentation

BP #129

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

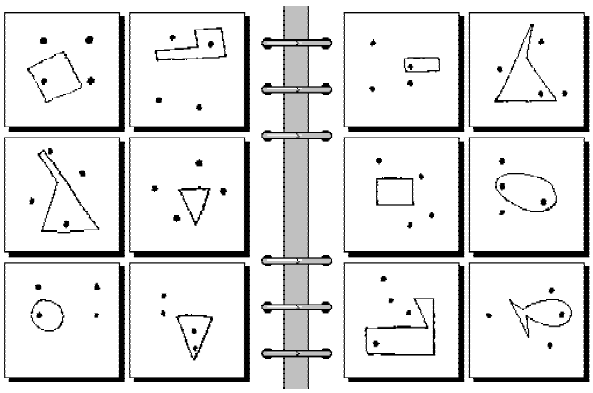


Larger closed region is made of curves

Larger closed region is made of line segments

BP #130

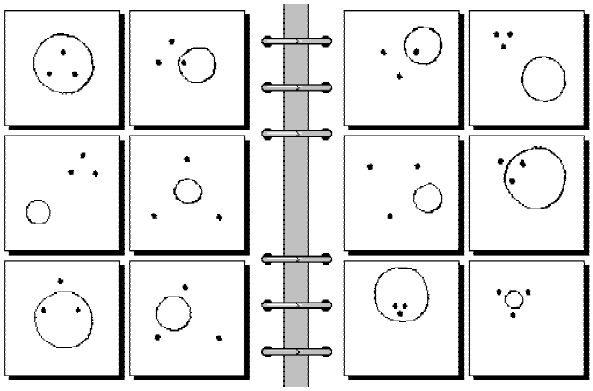
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Dots make up parallelogram

BP #131

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

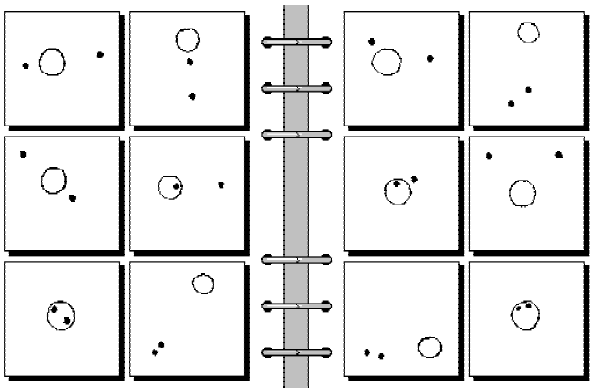


Dots make triangle with base down

Dots make triangle with vertex down

BP #132

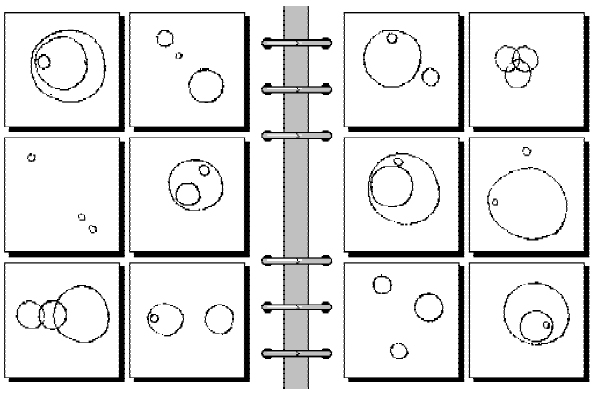
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Dots collinear with center of circle

BP #133

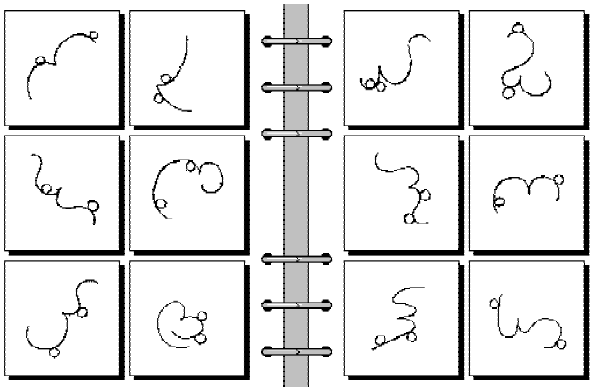
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Circle centers collinear

BP #134

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

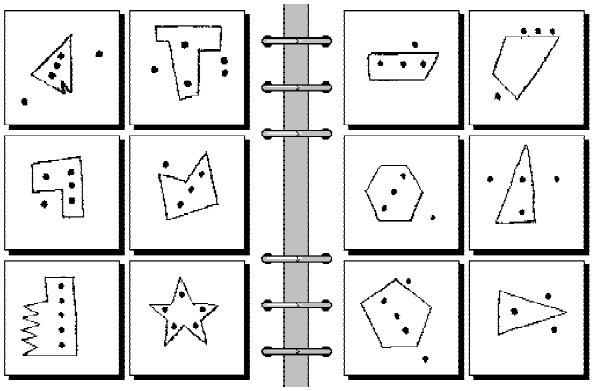


Circles on same side of curve

Circles on different sides of curve

BP #135

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

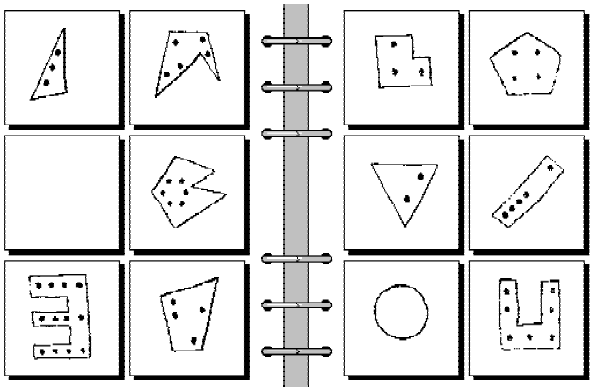


Concave shape

Convex shape

BP #136

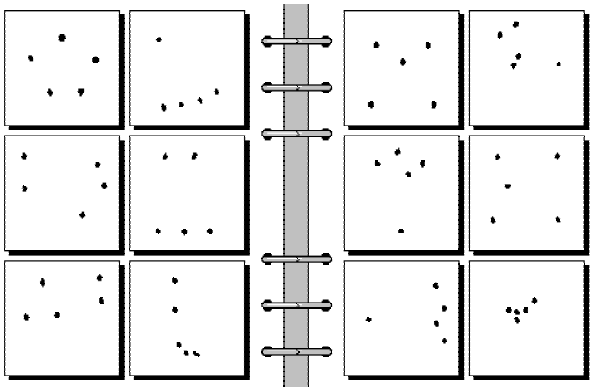
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Dots equal to the sides that make up the closed region

BP #137

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

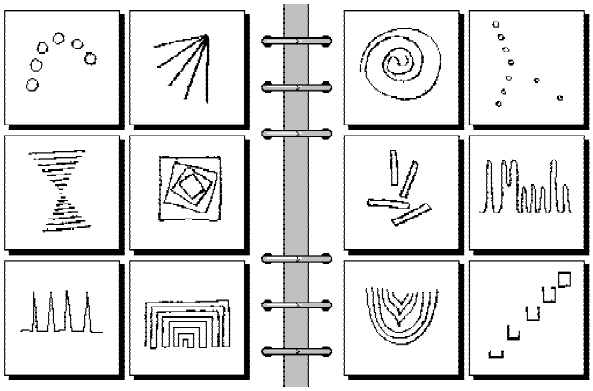


No dot in convex hull

At least one dot in convex hull

BP #138

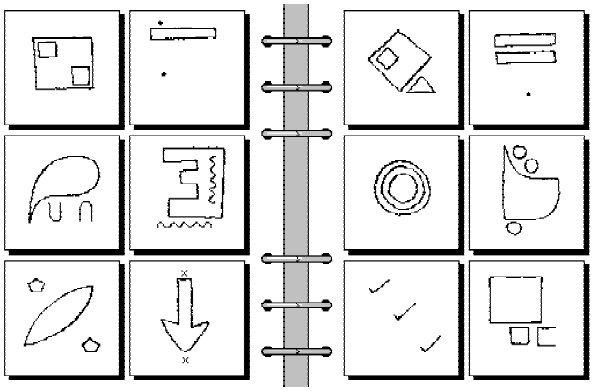
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Similar components that change regularly

BP #139

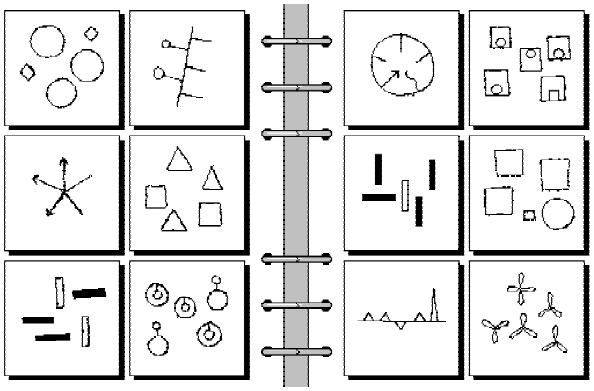
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



One large shape and two smaller identical ones

BP #140

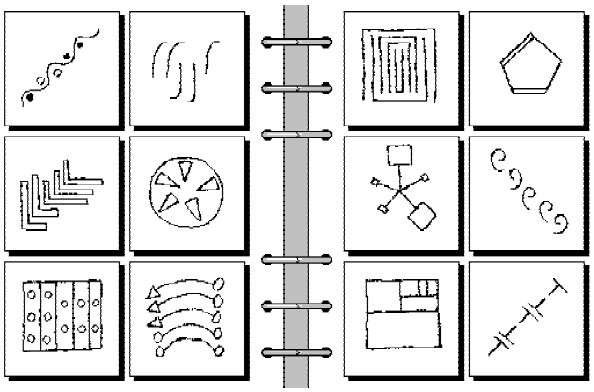
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Two groups of three and two

BP #141

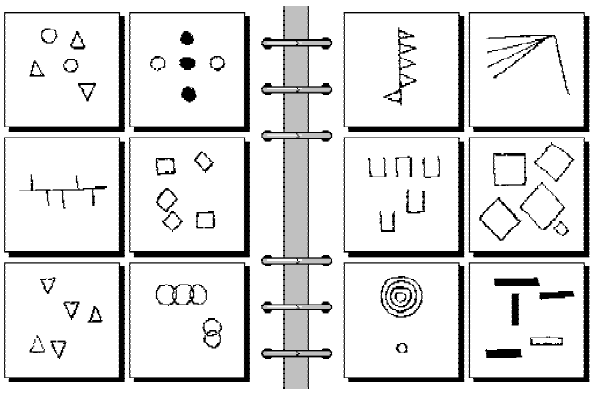
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Three and two, the two are always together

BP #142

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

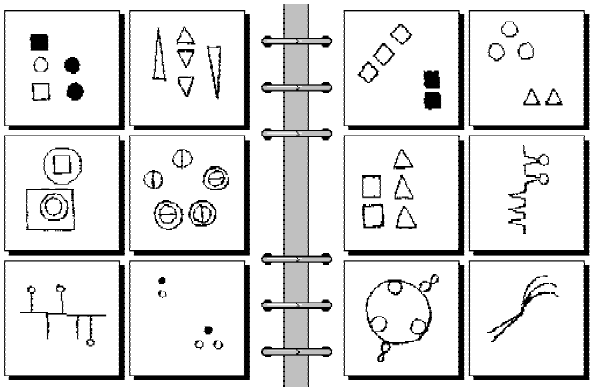


Three and two

Four and one

BP #143

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

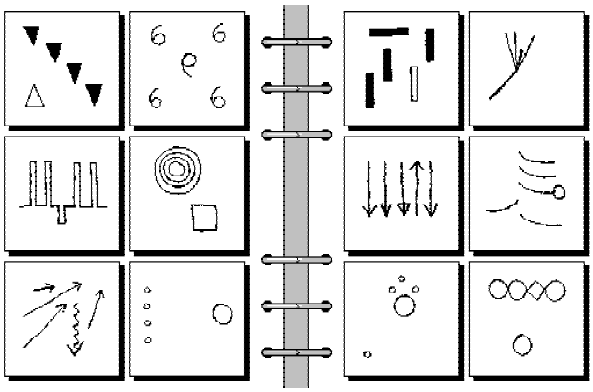


Three and two, but sharing a property

Three and two, but separable

BP #144

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

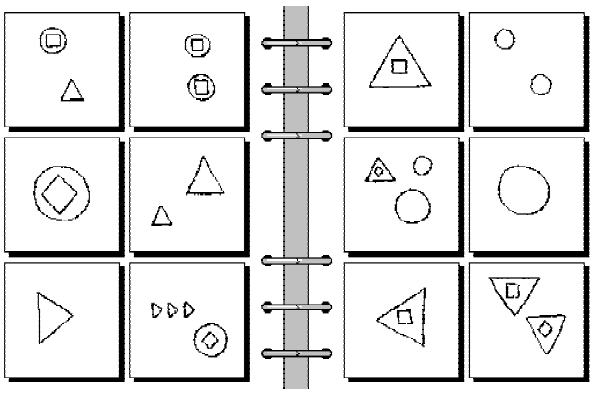


Four and one, the four make a regular group

Four and one, the four are three and one

BP #145

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

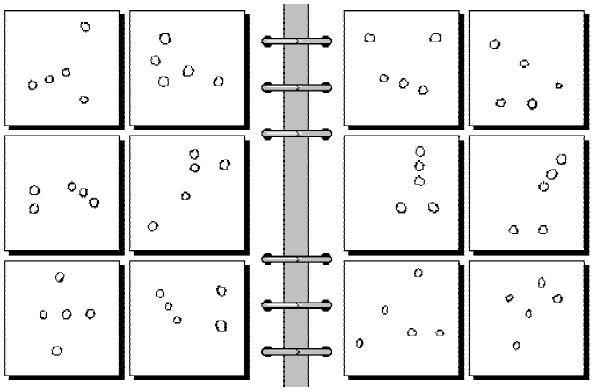


A square enclosed in a circle, and triangles

A square enclosed in a triangle, and circles

BP #146

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

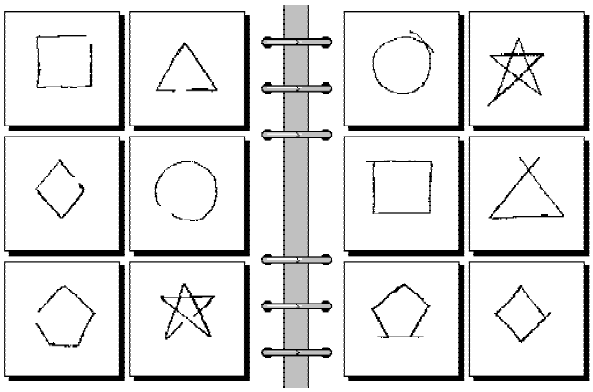


Three and two, the two are vertical

Three and two, the two are horizontal

BP #147

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

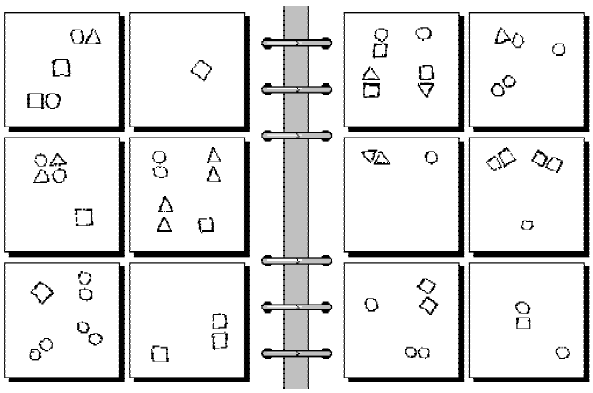


A little less than a regular shape

A little more than a regular shape

BP #148

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

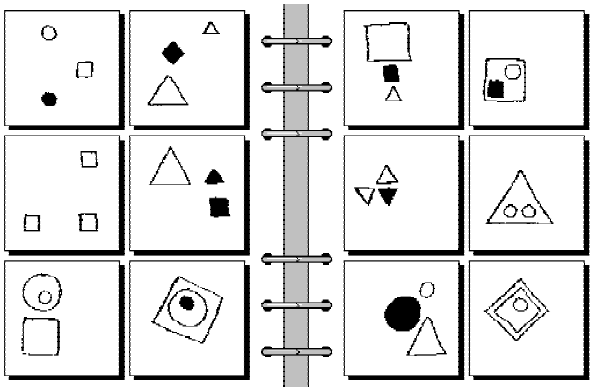


Lone square

Lone circle

BP #149

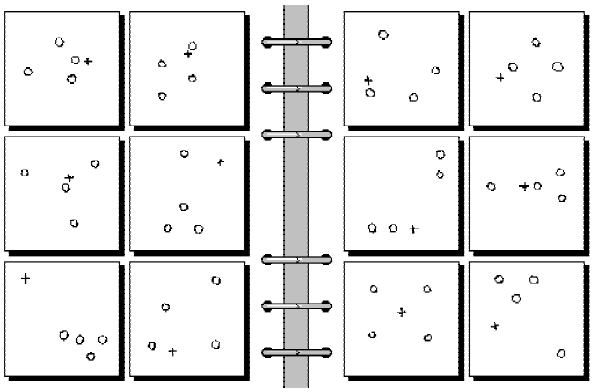
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Odd num. of squares Even num. of squares

BP #150

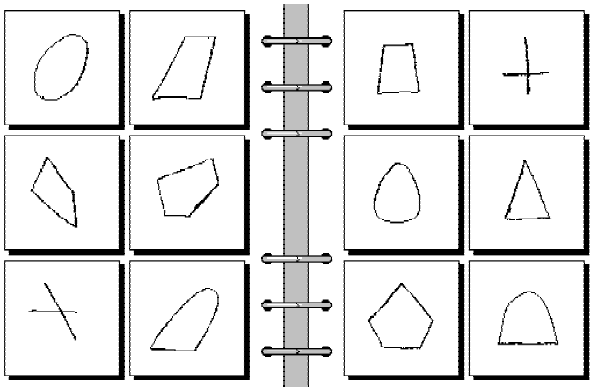
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



If the circle closest to the cross is removed,
the other three form an equilateral triangle

BP #151

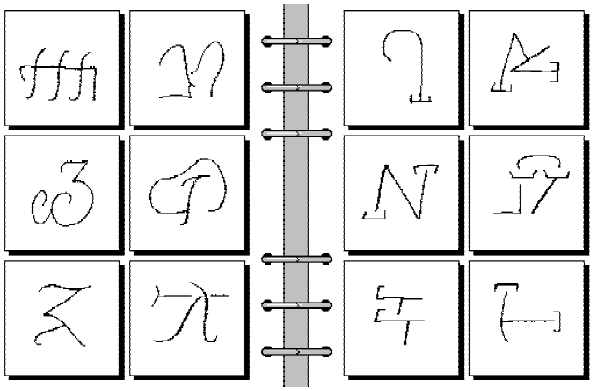
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Not vertically symmetric Vertically symmetric

BP #152

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

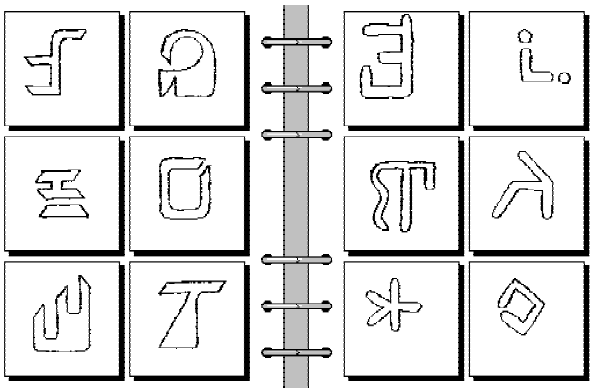


Hook-like ending

Square-bracket-like ending

BP #153

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

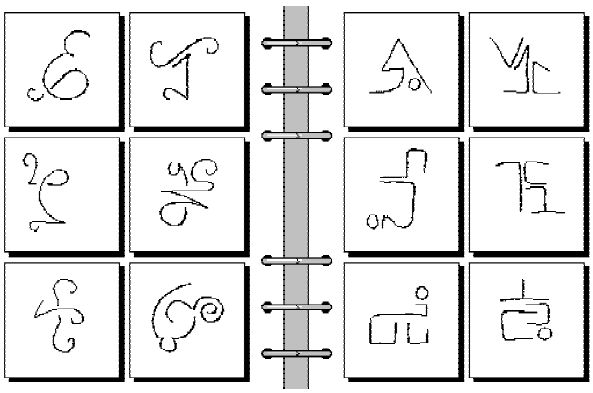


Wedged ending

Round ending

BP #154

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

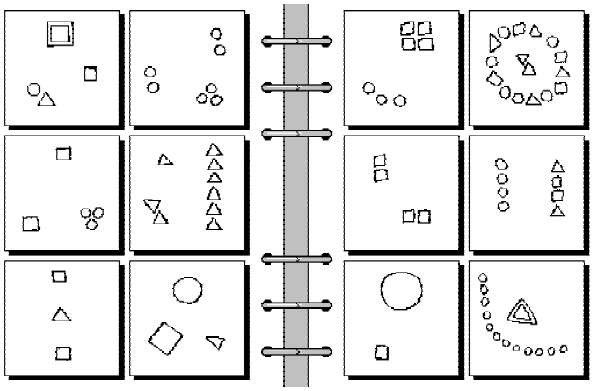


Curvaceous

Angular

BP #155

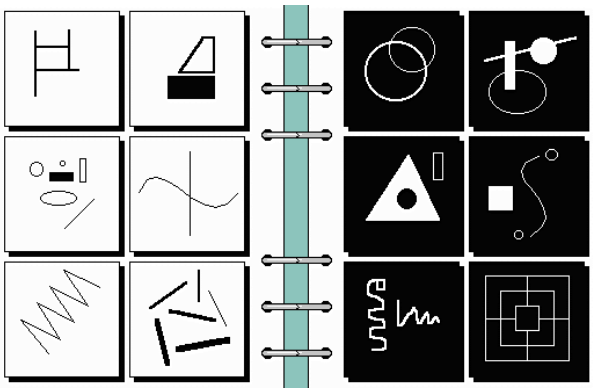
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Three groups Two groups

BP #156

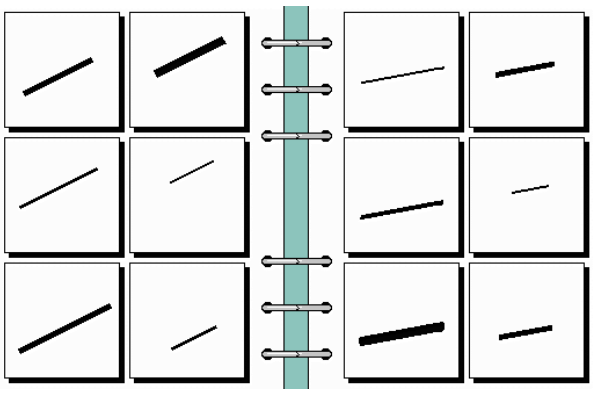
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



White background Black background

BP #157

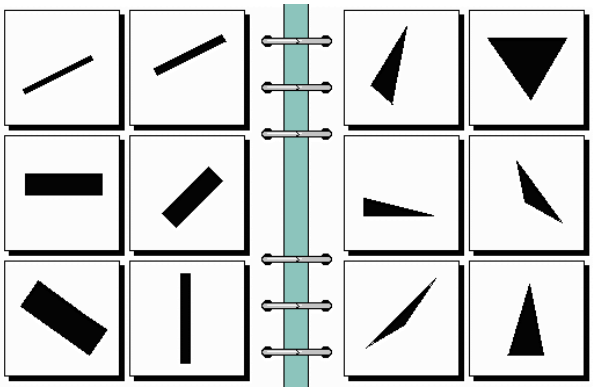
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Some slope Another slope

BP #158

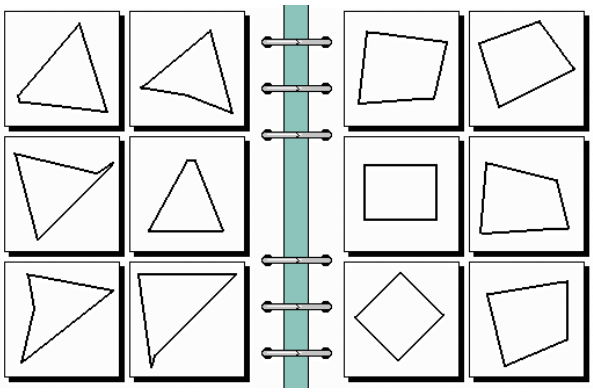
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Rectangle Triangle

BP #159

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

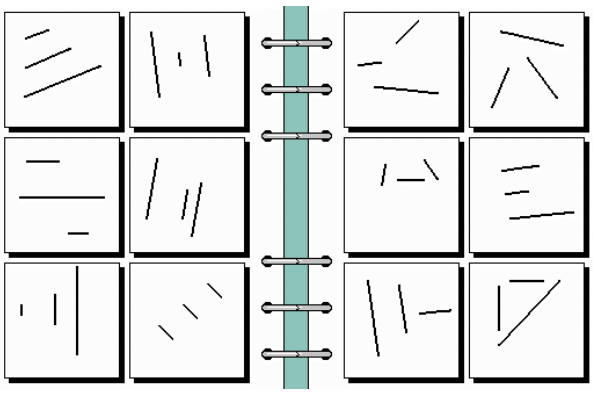


Rectangle which is nearly a triangle

Typical rectangle

BP #160

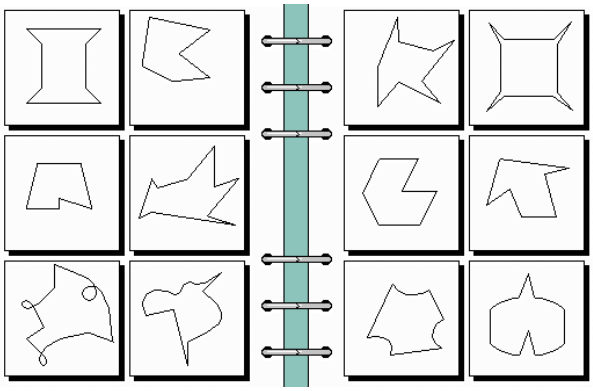
Performance:	*Human*	*Phaeaco*
Correct	26	
Avg. time	*23.2*	
Std. dev.	*23.9*	
Incorrect	1	
Avg. time	*65.0*	
No answer	1	
Avg. time	*122.0*	
Sample size	28	



Midpoints are collinear

BP #161

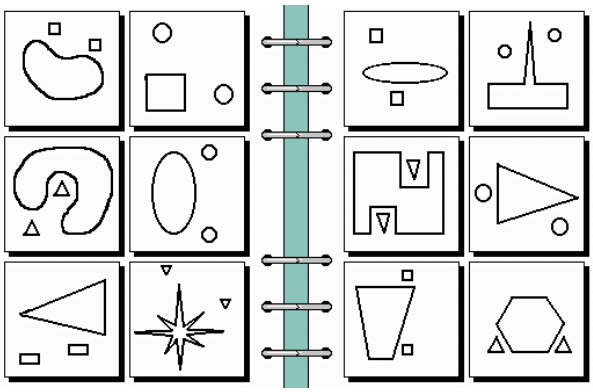
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Every other side passes through the same point

BP #162

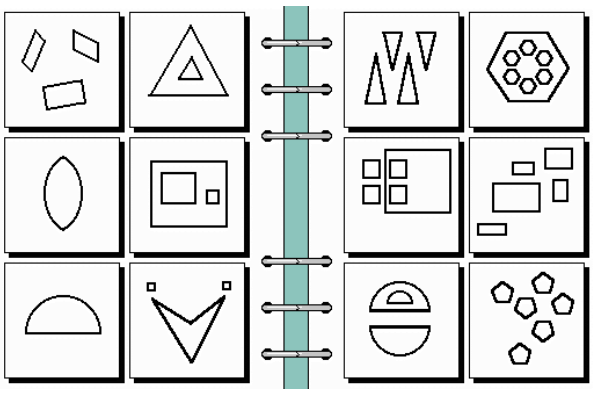
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Line connecting small objects does not intersect larger object

BP #163

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

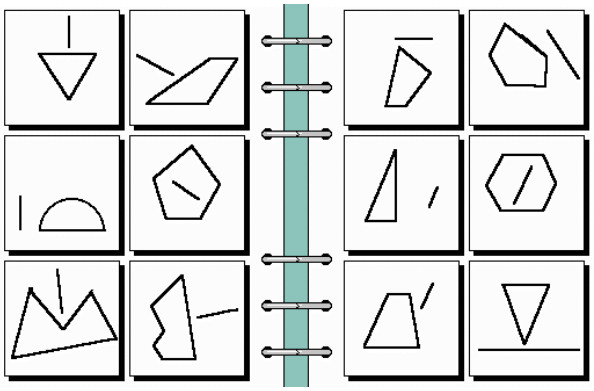


Number of objects is one less than sides

Number of objects is one more than sides

BP #164

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

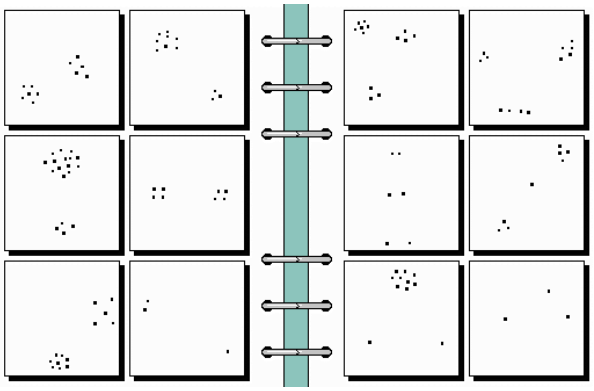


Line perpendicular to one side of the object

Line parallel to one side of the object

BP #165

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

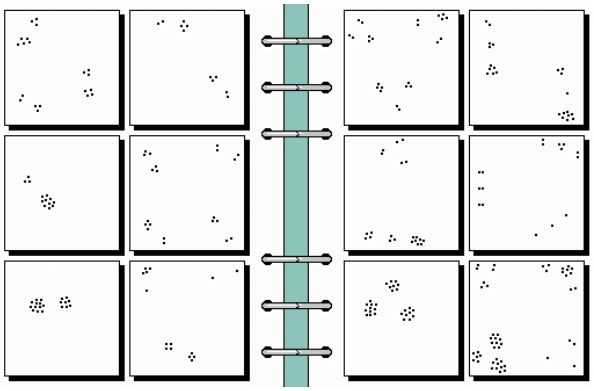


Two groups of dots

Three groups of dots

BP #166

Performance:	*Human*	*Phaeaco*
Correct	24	
Avg. time	*13.8*	
Std. dev.	*6.5*	
Incorrect	1	
Avg. time	*22.0*	
No answer	2	
Avg. time	*39.0*	
Sample size	27	

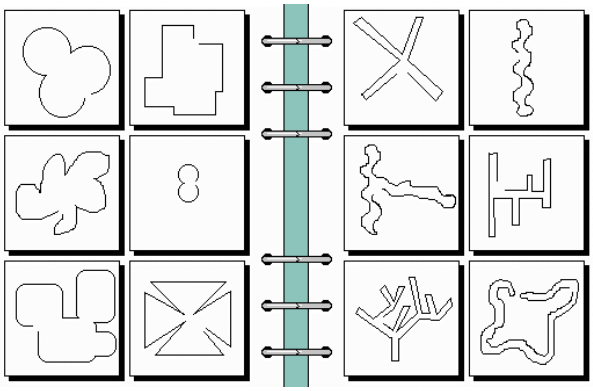


Groups of two groups of dots

Groups of three groups of dots

BP #167

Performance:	*Human*	*Phaeaco*
Correct	6	
Avg. time	*32.3*	
Std. dev.	*11.4*	
Incorrect	4	
Avg. time	*77.5*	
No answer	16	
Avg. time	*40.1*	
Sample size	26	

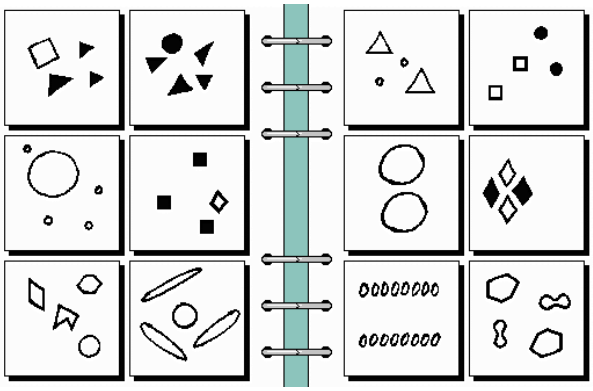


Bulky interior, if closed

Narrow interior, if closed

BP #168

Performance:	*Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*		
Incorrect *Avg. time*		
No answer *Avg. time*		
Sample size		

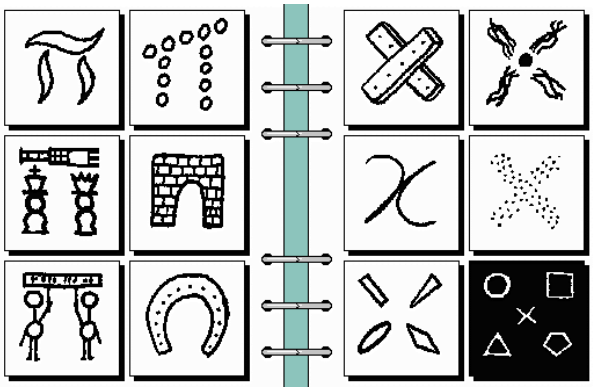


Odd

Even

BP #169

Performance:	*Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*		
Incorrect *Avg. time*		
No answer *Avg. time*		
Sample size		

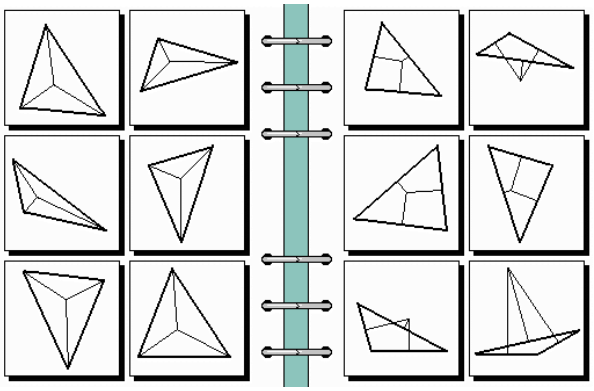


Π-like

X-like

BP #170

Performance:	*Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*		
Incorrect *Avg. time*		
No answer *Avg. time*		
Sample size		

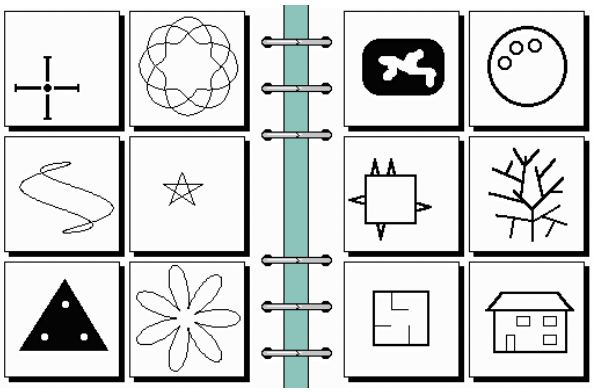


Angle bisectors meet at the incenter

Perpendicular bisectors meet at the orthocenter

BP #171

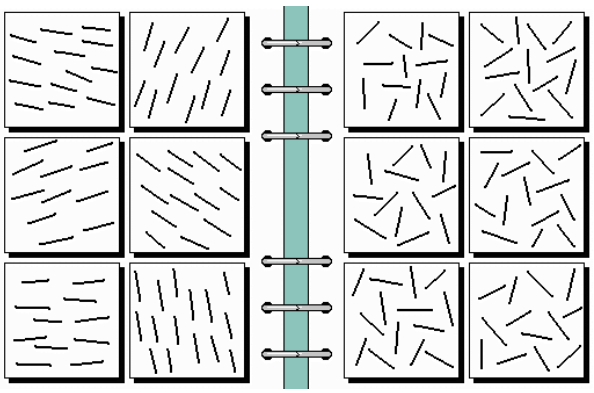
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Radially symmetric

BP #172

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

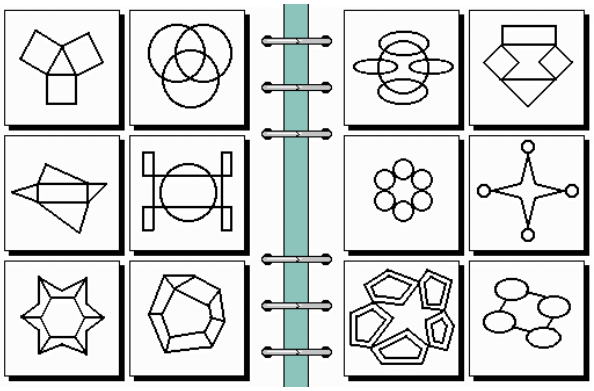


Small variance of slopes

Large variance of slopes

BP #173

Performance:	*Human*	*Phaeaco*
Correct	26	
Avg. time	*23.7*	
Std. dev.	*17.7*	
Incorrect	0	
Avg. time		
No answer	1	
Avg. time	*16.0*	
Sample size	27	

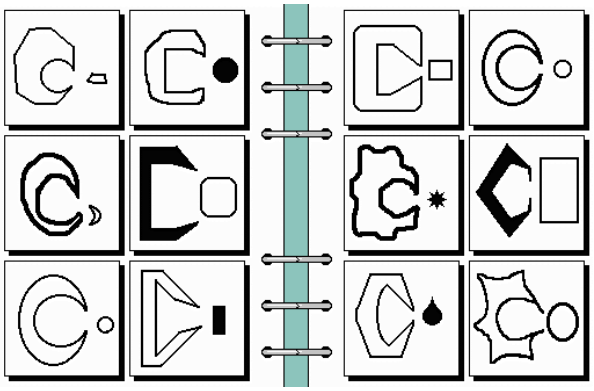


Convex central interior

Concave central interior

BP #174

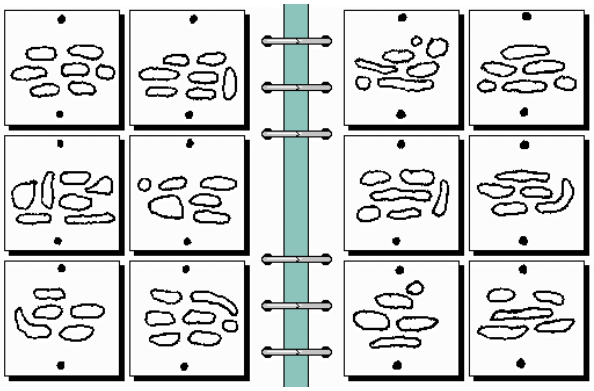
Performance:	*Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*		
Incorrect *Avg. time*		
No answer *Avg. time*		
Sample size		



Small object can glide in the bay

BP #175

Performance:	*Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*		
Incorrect *Avg. time*		
No answer *Avg. time*		
Sample size		

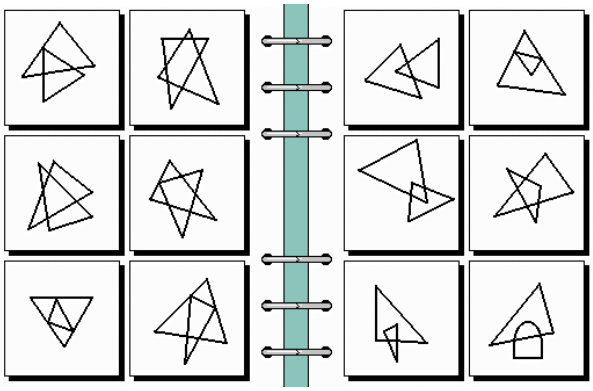


Short line connecting dots avoiding obstacles

Long line connecting dots avoiding obstacles

BP #176

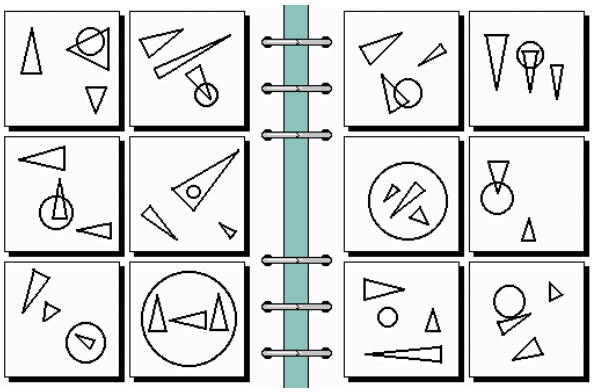
Performance:	*Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*		
Incorrect *Avg. time*		
No answer *Avg. time*		
Sample size		



All interiors are convex

BP #177

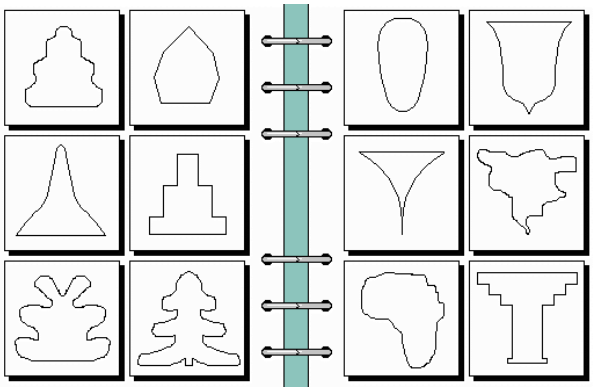
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Center of circle in triangle
perpendicular to the other two

BP #178

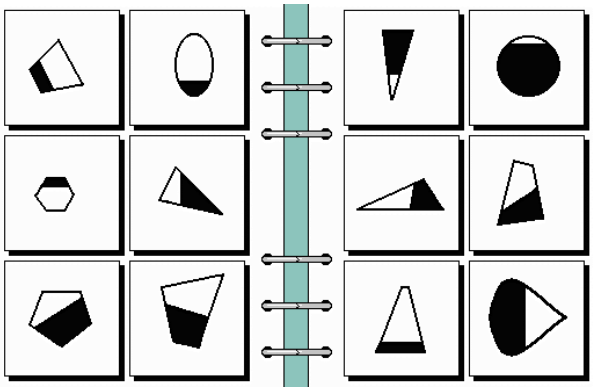
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Thinner at top Thicker at top

BP #179

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

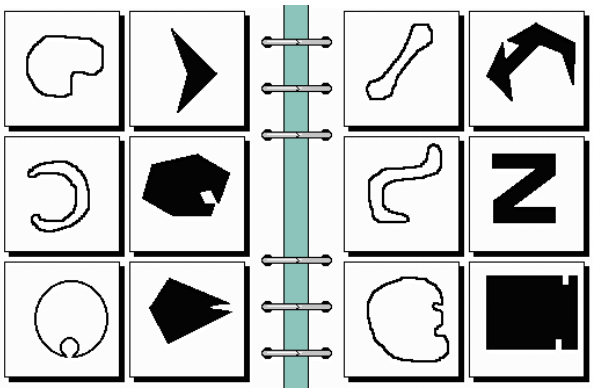


Black region widens toward the center

Black region narrows toward the center

BP #180

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

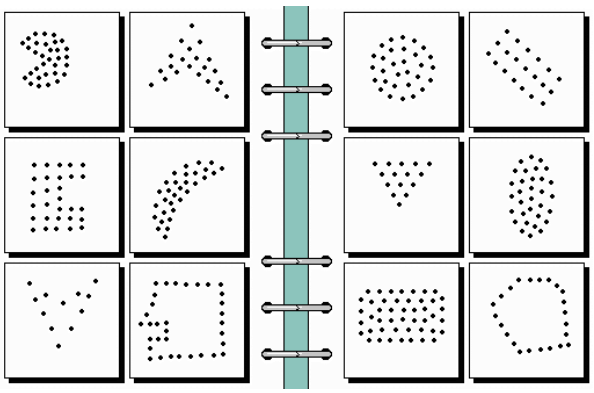


One concavity

Two concavities

BP #181

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

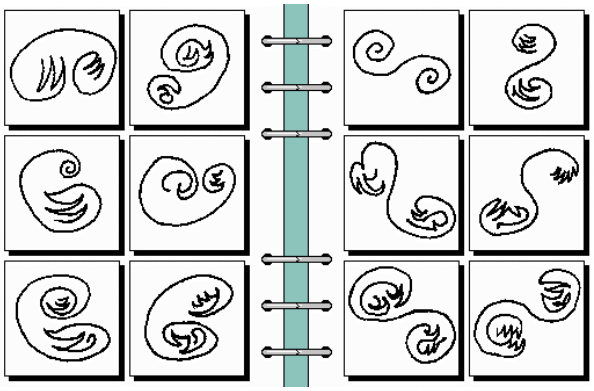


Concave if proximal points are connected

Convex if proximal points are connected

BP #182

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

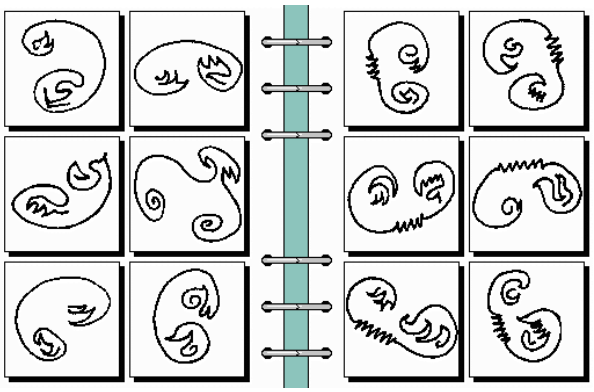


Same curvature close to the middle

Change of curvature close to the middle

BP #183

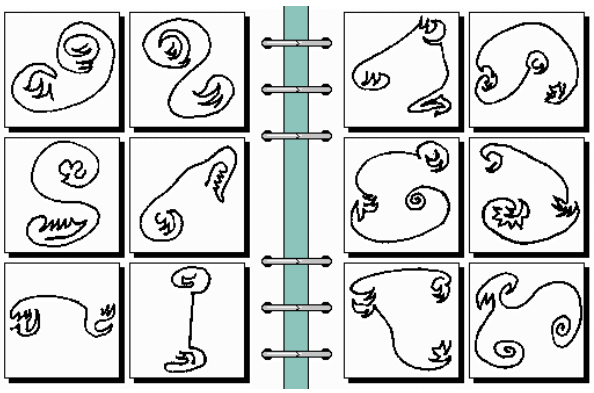
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Zigzag part close to the middle

BP #184

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

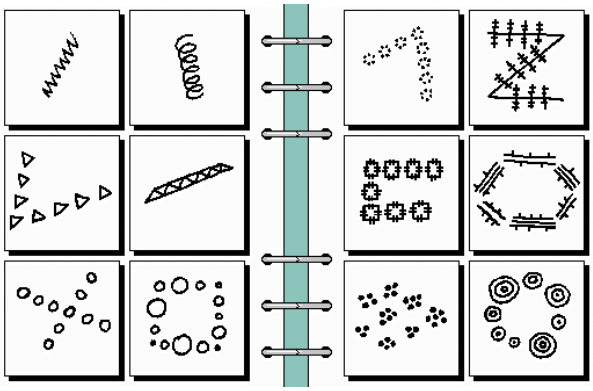


Two complex parts

Three complex parts

BP #185

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

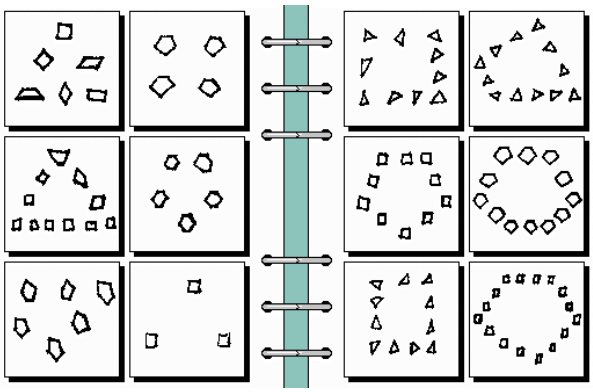


Object made of objects

Object made of objects made of objects

BP #186

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

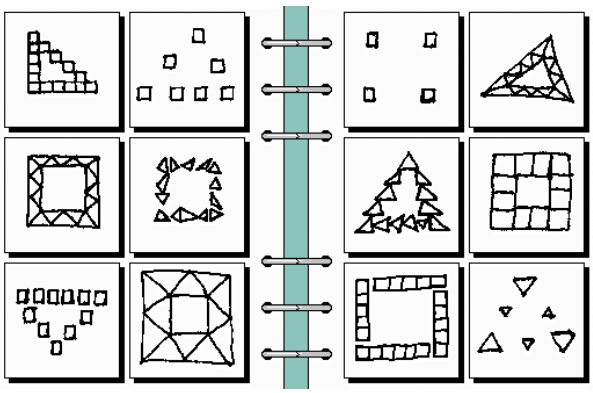


Sides of parts one more than sides of whole

Sides of parts one less than sides of whole

BP #187

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

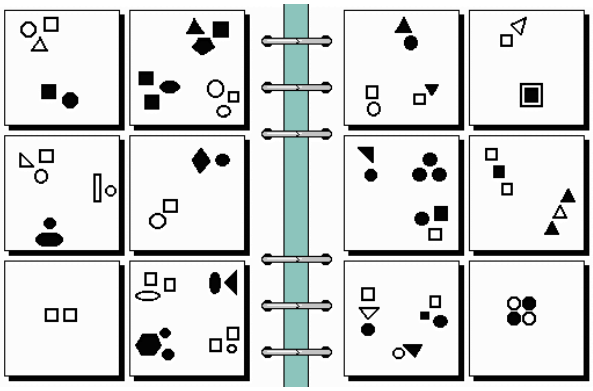


Shape of whole different from shape of parts

Shape of whole same as shape of parts

BP #188

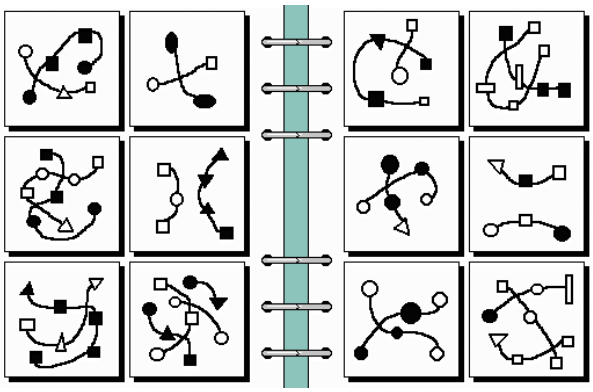
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



All groups are made of parts of the same texture

BP #189

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

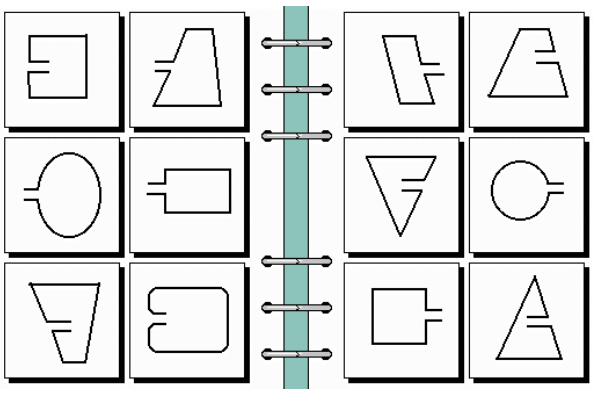


All connected pieces have the same texture

Some connected pieces have different textures

BP #190

Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		

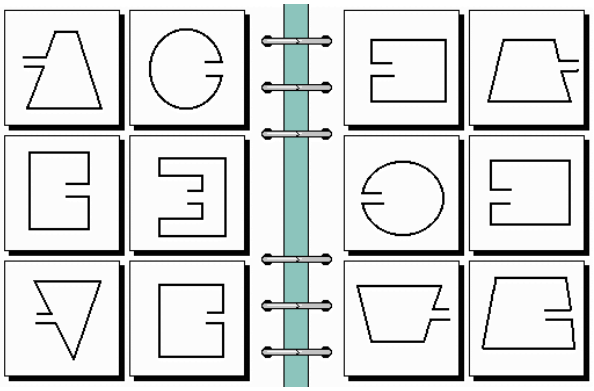


Orifice on the left

Orifice on the right

BP #191

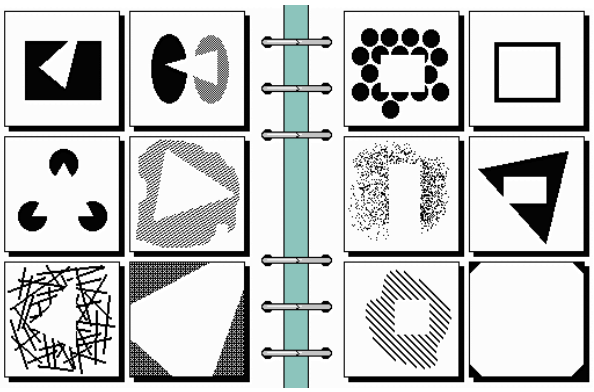
Performance:	*Human*	*Phaeaco*
Correct		
Avg. time		
Std. dev.		
Incorrect		
Avg. time		
No answer		
Avg. time		
Sample size		



Elongated vertically Elongated horizontally

BP #192

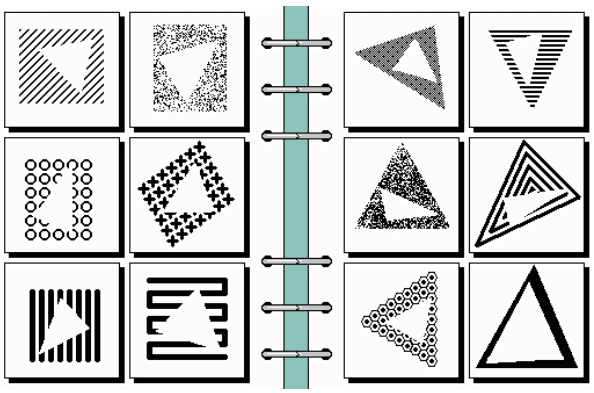
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Triangle Rectangle

BP #193

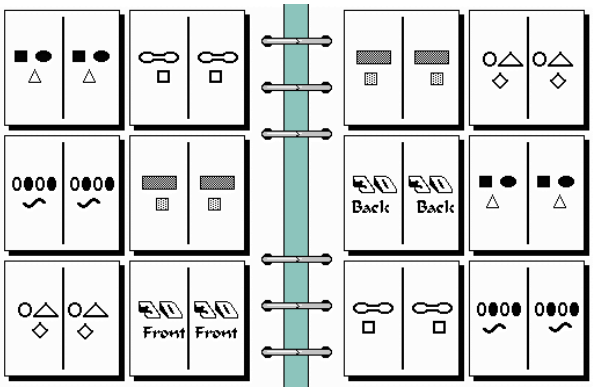
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Quadrilateral background Triangular background

BP #194

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

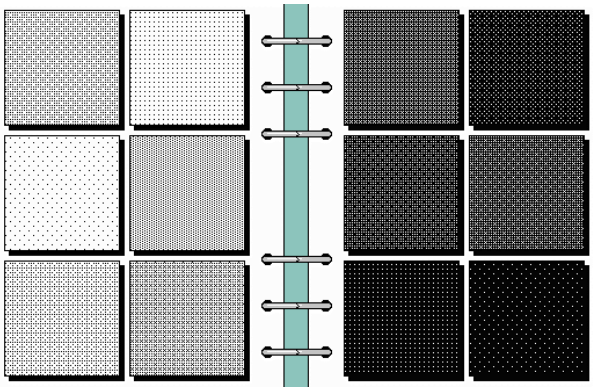


Bottom object in front of top objects in 3-D

Bottom object behind top objects in 3-D

BP #195

Performance:	*Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*		
Incorrect *Avg. time*		
No answer *Avg. time*		
Sample size		

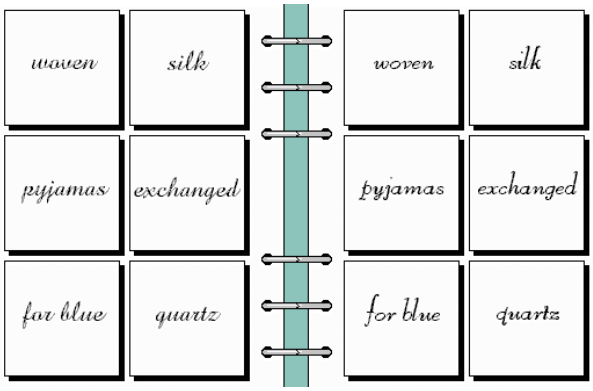


Light-colored texture

Dark-colored texture

BP #196

Performance:	*Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*		
Incorrect *Avg. time*		
No answer *Avg. time*		
Sample size		

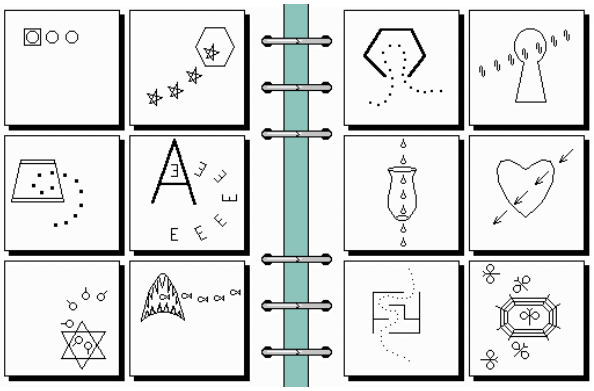


Some style (font)

Another style (font)

BP #197

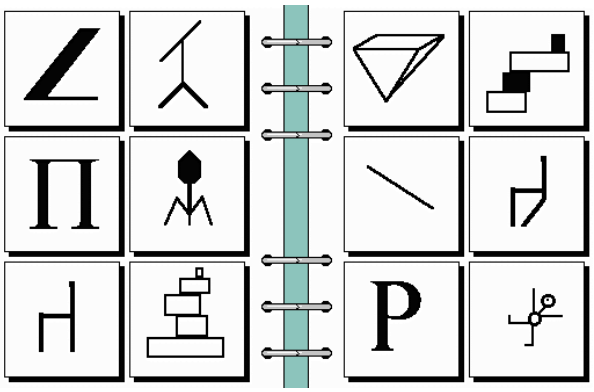
Performance:	*Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*		
Incorrect *Avg. time*		
No answer *Avg. time*		
Sample size		



Stays in Escapes

BP #198

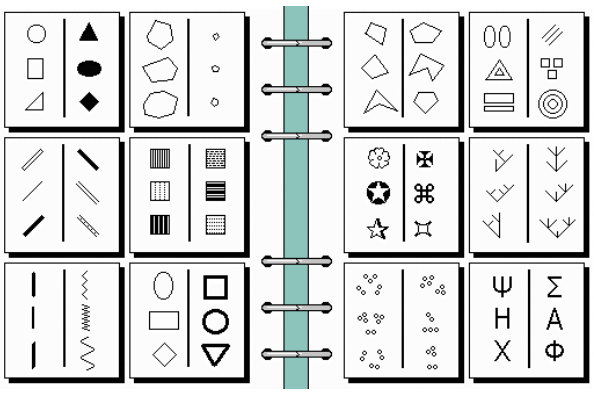
Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Stays put Tumbles

BP #199

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	



Solution of BP is based on featural difference

Solution of BP is based on numerosity

BP #200

Performance: *Human*	*Phaeaco*
Correct *Avg. time* *Std. dev.*	
Incorrect *Avg. time*	
No answer *Avg. time*	
Sample size	

Appendix B: Curve Approximation

1. Parametric cubic b-splines

Given are $n + 1$ points, P_0, P_1, …, P_n, on the 2-D Euclidean plane. The task is to find n 3rd-degree (cubic) polynomials S_0, S_1, …, S_{n-1}, such that each S_i passes through points P_i and P_{i+1}, and the overall curve formed by the cubic polynomials is smooth (Figure B.1). This last condition implies that the first and second derivatives of polynomials S_{i-1} and S_i at point P_i are equal.

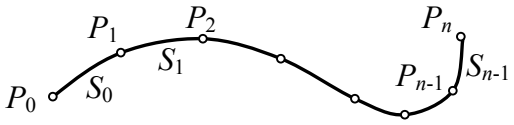


Figure B.1: n +1 points approximated by a piecewise smooth curve

The polynomials S_i , $i = 0,\ldots,n\text{-}1$, will be expressed in parametric equations:

$$S_i(t) = [a_i(t), b_i(t)],\ 0 \le t \le 1,\ \text{where}$$

$$[a_i(0), b_i(0)] = P_i, \text{ and}$$

$$[a_i(1), b_i(1)] = P_{i+1}.$$

Denote polynomial $a_i(t)$ by $a_{3,i}t^3 + a_{2,i}t^2 + a_{1,i}t + a_{0,i}$, and, similarly, polynomial $b_i(t)$ by $b_{3,i}t^3 + b_{2,i}t^2 + b_{1,i}t + b_{0,i}$. Thus, the derivatives of S_i are:

$$S_i'(t) = [3a_{3,i}t^2 + 2a_{2,i}t + a_{1,i}\ ,\quad 3b_{3,i}t^2 + 2b_{2,i}t + b_{1,i}\ ], \text{ and}$$

$$S_i''(t) = [3a_{3,i}t + a_{2,i}\ ,\quad 3b_{3,i}t + b_{2,i}\ ]$$

(Note: The terms in the second derivative were simplified, divided by 2.) In what follows, only the derivation of $a_i(t) = a_{3,i}t^3 + a_{2,i}t^2 + a_{1,i}t + a_{0,i}$ is shown, since the derivation of $b_i(t)$ is exactly symmetrical (i.e., replace a by b).

The requirements for piecewise smooth curves translate into the following:

$S_i(1) = S_{i+1}(0) = P_{i+1}$, **(1)**

$S_i'(1) = [a_i'(1), b_i'(1)] = [a_{i+1}'(0), b_{i+1}'(0)] = S_{i+1}'(0)$, **(2)** and

$S_i''(1) = [a_i''(1), b_i''(1)] = [a_{i+1}''(0), b_{i+1}''(0)] = S_{i+1}''(0)$ **(3)**.

From **(1)** we have: $S_i(0) = P_i$, hence: $a_{0,i} = x_i$ **(4)** (where x_i is the x-coordinate of point P_i), and also: $S_i(1) = P_{i+1}$, hence: $a_{3,i} + a_{2,i} + a_{1,i} + a_{0,i} = x_{i+1}$ **(5)**.

From **(2)** we have: $3a_{3,i} + 2a_{2,i} + a_{1,i} = a_{1,i+1}$ **(6)**.

From **(3)** we have: $3a_{3,i} + a_{2,i} = a_{2,i+1} \Rightarrow a_{3,i} = \dfrac{a_{2,i+1} - a_{2,i}}{3}$ **(7)**.

From **(7)** and **(6)** we obtain: $a_{1,i} = a_{2,i} + a_{2,i-1} + a_{1,i-1}$ **(8)**.

From **(7)** and **(5)** we obtain:

$\dfrac{a_{2,i+1} - a_{2,i}}{3} + a_{2,i} + a_{1,i} + a_{0,i} = x_{i+1}$, and using **(4)** the last equation becomes:

$a_{2,i+1} - a_{2,i} + 3a_{2,i} + 3a_{1,i} = 3(x_{i+1} - x_i)$, from which the following two equations are obtained:

$a_{1,i} = x_{i+1} - x_i - \dfrac{1}{3}(2a_{2,i} + a_{2,i+1})$ **(9a)**, and

$a_{1,i-1} = x_i - x_{i-1} - \dfrac{1}{3}(2a_{2,i-1} + a_{2,i})$ **(9b)**.

From **(8)**, **(9a)**, and **(9b)**, we obtain:

$$a_{2,i} + a_{2,i-1} + x_i - x_{i-1} - \frac{1}{3}(2a_{2,i-1} + a_{2,i}) = x_{i+1} - x_i - \frac{1}{3}(2a_{2,i} + a_{2,i+1}) \Rightarrow$$

$$\frac{1}{3}a_{2,i-1} + \frac{4}{3}a_{2,i} + \frac{1}{3}a_{2,i+1} = x_{i+1} - x_i - (x_i - x_{i-1}) \Rightarrow$$

$a_{2,i-1} + 4a_{2,i} + a_{2,i+1} = 3x_{i-1} - 6x_i + 3x_{i+1}$ **(10)**.

Equation **(10)** holds for $i = 1, \ldots, n-1$, i.e., it yields $n-1$ equations. To obtain two more equations (because there are $n+1$ unknowns), we can impose the following two boundary conditions:

$S_0''(0) = 0 \Rightarrow a_{2,0} = 0$ **(11)**, and

$S_{n-1}''(1) = 0 \Rightarrow 3a_{3,n-1} + a_{2,n-1} = 0$, whence using **(7)** we have:

$$3\frac{a_{2,n} - a_{2,n-1}}{3} + a_{2,n-1} = 0 \Rightarrow a_{2,n} - a_{2,n-1} + a_{2,n-1} = 0 \Rightarrow a_{2,n} = 0 \quad \textbf{(12)}.$$

Thus, equations **(10)**, **(11)**, and **(12)** result in a system of $n+1$ equations in $n+1$ unknowns (or rather, (n – 1) × (n – 1), since $a_{2,0} = a_{2,n} = 0$, hence the first and last rows and the first and last columns of the system are zero):

$$\left\langle \begin{array}{ccccccccc|c}
a_{2,0} & & & & & & & & & 0 \\
a_{2,0} & 4a_{2,1} & a_{2,2} & & & & & & & 3x_0 - 6x_1 + 3x_2 \\
 & a_{2,1} & 4a_{2,2} & a_{2,3} & & & & & & 3x_1 - 6x_2 + 3x_3 \\
 & & \cdots & \cdots & \cdots & & & & & \cdots \\
 & & & a_{2,i-1} & 4a_{2,i} & a_{2,i+1} & & & & 3x_{i-1} - 6x_i + 3x_{i+1} \\
 & & & & \cdots & \cdots & \cdots & & & \cdots \\
 & & & & & a_{2,n-2} & 4a_{2,n-1} & a_{2,n} & & 3x_{n-2} - 6x_{n-1} + 3x_n \\
 & & & & & & & a_{2,n} & & 0
\end{array} \right\rangle$$

The above is a tridiagonal system, and can be solved by using Gaussian elimination adapted for such systems (e.g., Press, Flannery *et al.*, 1986, p. 40). Once the $a_{2,i}$'s are obtained by solving the above, the $a_{1,i}$'s are computed from equation **(9a)**, whereas the $a_{3,i}$'s are computed from equation **(7)**. Finally, the $a_{0,i}$'s are already known from **(4)**. The coefficients of the polynomial $b_i(t)$ are computed in an exactly analogous way, using the y-coordinates of the points P_i.

2. Ellipse and circle detection

Given are n points, $P_1, \ldots, P_n$, $n \geq 6$, on the 2-D Euclidean plane. The task is to determine whether the points lie approximately on an elliptical arc in general, or circular arc in particular, and to estimate the coefficients of the equation of the curve. The general equation of a conic section on the plane is given below:

$$ax^2 + 2bxy + cy^2 + 2dx + 2fy + 1 = 0 \quad \textbf{(13)}$$

If $ac > 0$, then the conic section is an ellipse or circle. (If $ac < 0$, then it is a hyperbola or pair of intersecting straight lines, and if $ac = 0$ then it is a parabola). Clearly, five points suffice to estimate coefficients *a, b, c, d,* and *f* in **(13)**, since they yield a linear system of five equations in five unknowns, which generally has a unique solution. However, *any* five points (barring collinearities) can satisfy **(13)**. Thus, six or more points are required to guarantee that after solving the resulting $n \times 5$ linear system, $n \geq 6$, **(13)** describes a conic section.

After estimating *a, b, c, d,* and *f,* the points $P_1, \ldots, P_n$ can be plugged back into **(13)**, to determine how well they satisfy the equation. Specifically:

$$ax^2 + 2bxy + cy^2 + 2dx + 2fy = g \quad \textbf{(14)}$$

In **(14)**, each P_i, $i = 1, \ldots, n$, yields a value for g that must be approximately equal to –1. By collecting a sample of g_i, $i = 1, \ldots, n$, we can determine whether the points lie on a conic section by examining the mean value $\bar{x}$ of the sample, which must be near –1, and its standard deviation *s,* which must be near 0. In practice, it suffices that $|\bar{x} - (-1)| < 0.05$, and $s < 0.01$.

Assuming $ac > 0$, we have an ellipse or circle. The following computations result in an estimate of several parameters of the ellipse.

$$k = \frac{c-a}{2b}, \quad l = \frac{k}{\sqrt{1+k^2}}, \quad \sin\theta = \sqrt{\frac{1+l}{2}}, \quad \cos\theta = \sqrt{\frac{1-l}{2}} \quad \textbf{(15)}$$

In equations **(15)**, θ is the angle by which the ellipse must be turned in order to be either horizontally or vertically oriented. The resulting rotated ellipse has the following coefficients:

$$a' = a\cos^2\theta - 2b\cos\theta\sin\theta + c\sin^2\theta \qquad \textbf{(16)}$$

$$b' = b\left(\cos^2\theta - \sin^2\theta\right) + (a - c)\sin\theta\cos\theta \qquad \textbf{(17)}$$

$$c' = a\sin^2\theta + 2b\sin\theta\cos\theta + c\cos^2\theta \qquad \textbf{(18)}$$

$$d' = d\cos\theta - f\sin\theta$$

$$f' = f\cos\theta - d\sin\theta$$

The value of b' in **(17)** bust be approximately equal to zero, since the axes of the rotated ellipse are parallel to the x and y axes. Equation **(19)**, below, yields an estimate of the *eccentricity* E of the ellipse:

$$\text{if } |a'| > |c'| \text{ then } E = \sqrt{1 - \frac{c'}{a'}}, \text{ else } E = \sqrt{1 - \frac{a'}{c'}} \qquad \textbf{(19)}$$

Ideally, the value of E in (19) is zero if the points form a circle. But in practice the points are computed approximately only (in Phaeaco they are integers), so any value of E below around 0.45 suggests a circle. The larger the circle, the more this threshold can be "trusted" (i.e., the lower its value can be). Circles that nearly fill Phaeaco's visual box (100×100) yield an eccentricity of around 0.20.

3. Computation of parameters of a circle

Given are n points, P_1, …, P_n, $n \geq 6$, on the 2-D Euclidean plane that are suspected to lie on a circle. To compute the center of the circle, sample triplets of points $[P_{i_1}, P_{i_2}, P_{i_3}]$ taken from among the n points can be examined, and the circumcenter (C_{i_x}, C_{i_y}) of the triangle formed by each triplet can be computed. If the n points lie on a circle, the circumcenters must approximately coincide.

Therefore, a method for testing how well the given points form a circle is to examine the standard deviation of each of the samples of x- and y-coordinates of the circumcenters, and demand that it is a small number, close to zero. The circumcenter of a triangle $P_1P_2P_3$ where $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, and $P_3 = (x_3, y_3)$, is given by the formulas:

$$C_x = \frac{y_3 - y_2 + \dfrac{(x_3 - x_1)(x_3 + x_1)}{y_3 - y_1} - \dfrac{(x_2 - x_1)(x_2 + x_1)}{y_2 - y_1}}{2\left(\dfrac{x_3 - x_1}{y_3 - y_1} - \dfrac{x_2 - x_1}{y_2 - y_1}\right)}$$

$$C_y = -C_x \frac{x_3 - x_1}{y_3 - y_1} + \frac{y_3 + y_1}{2} + \frac{(x_3 - x_1)(x_3 + x_1)}{2(y_3 - y_1)}$$

The above formulas are valid provided that $y_1 \neq y_2$ and $y_1 \neq y_3$. But note that since the three points form a triangle, hence are not collinear, at least one of $y_1 \neq y_2$ or $y_1 \neq y_3$ must be true. If any (but not both) of $y_1 = y_2$ and $y_1 = y_3$ is true, then renaming the points appropriately makes the above formulas usable.

Having computed the center of the circle, the radius is found as the Euclidean distance between the center and any of the n points. Taking the average of all n distances yields a better estimate of the radius.

Appendix C: Origin of Phaeaco's name

There are several reasons — all listed below — for which I chose the name "Phaeaco" (/fi·'ɑ·ko/) for the system described in this thesis.

1. It is a proper name

There is a passage in Homer's *Odyssey* that I find extremely interesting from a modern, cognitive-science perspective. In Books Eta and Theta, the *Phaeacians* are described: a peace-loving and sea-faring people, who probably lived on an island that in our times is called Corfu (just off the northwestern coast of Greece; its shape has an irregular *elongatedness*). The Phaeacians are the first to welcome in their land the hero, *Odysseus* (in Latin: Ulysses), who has suffered a long and arduous journey over the seas. The Phaeacian king *Alcinous* orders his people to construct a ship for Odysseus to help him travel to his final destination, the island of *Ithaca,* where his home is. Alcinous, while describing to Odysseus the virtues of the Phaeacian ships, says the following (translation by the author, with some vital help from Douglas Hofstadter and Kellie Gutman):

"Your country, tell me what it's called, your people and your city;
then let our ships design their course, and bring you safely homeward.
Phaeacian ships do not have men to steer them on their courses;
they lack the rudders you may see in ships of other nations;
but on their own they guess the thoughts and wishes of their makers;
they know all countries of the world, their cities, and their meadows;
they travel swiftly like the wind that blows o'er seas and oceans,
avoiding storms and cloudy skies, so they are not in danger
of sailing off their course to founder, sink, and slowly vanish."

Who said Artificial Intelligence is a modern concept? Nearly 3,000 years ago it seems Homer had some grasp of it. I found this image very interesting: a program-ship *that on its own may guess the thoughts and wishes of its maker*, and sail through misty cognitive spaces to find, unharmed, its target.

The above excerpt is from Book Theta, two paragraphs before the end. Earlier, in Book Eta, goddess *Athena* (in Latin: Minerva), in disguise, tells Odysseus the following about the Phaeacians:

"They're sailors, that is what they are, whose ships, by Neptune's graces,
glide o'er the seas like birds, or like perceptions through your spirit."

It appears Homer had some cognitive project in his mind! Now, if only he had given the Phaeacian ship a name! Well, unfortunately, he did not. So I decided to give it a name myself. I thought the name should reflect the ship's origin, and should be of feminine gender (as all ancient names of ships were), rhyming with another important ancient ship's name.

2. It rhymes (in Greek) with Argo

Argo was the name of the ship of another (non-Homeric) hero: *Jason,* who sailed the seas to find the golden fleece, helped by his comrades, the *Argonauts.* Since I am one of the "Fargonauts" (from FARG, our research group), I thought that's cute, too. Here are the names of the two ships, in the original language:

Αργώ – Φαιακώ

3. It is an acronym in English

Once, my research advisor remarked, "I find most of today's acronyms very contrived." I kept trying to find an acronym for my project for months, but could

not come up with one that would both satisfy my advisor and conform to the spirit of our research group's prior names for projects. So I gave up and decided instead that I would devise an acronym that would be obviously *too contrived*. Eventually I succeeded, coming up with the following monstrosity:

P = Pattern-recognizing (an allusion to the title of M. M. Bongard's book)
H = Hofstadter-inspired (a tribute to my advisor)
A = Architecture (that's what Phaeaco is)
E = Empirically (its justification comes from empirical observations)
A = Approaching (it is just an approximation; I hope it will keep approaching...)
C = Cognitive
O = Organization

Is this sufficiently *contrived*? But wait... there is more!

4. It is an acronym in Greek, too!

Here it is: ΦΑΙΑΚΩ, in ancient (Attic) Greek, makes up the following acronym:

Φ = Φουνταλῆς
Α = ἀποπλεῦσας
Ι = ᾽Ιθάκηξεν
Α = ἀλκίνοον
Κ = κάραν
Ω = ὡμοίωσεν

Now, what does all that *mean*? I leave this as an exercise for the reader. Suffice it to say that the words are not random, they *are* meaningful, make up a sentence in ancient Greek, and relate both to this project and to the Odyssey.

REFERENCES

Antell, Sue Ellen and Daniel P. Keating (1983). "Perception of numerical invariance in neonates". *Child Development*, no. 54, pp. 695-701.

Aristotle (1992). *Categories. On Interpretation. Prior Analytics*: Loeb Classical Library. Harvard.

Arms, Karen and Pamela S. Camp (1988). *Biology: A Journey into Life*: Saunders College Publishing.

Armstrong, S. L., Lila R. Gleitman*, et al.* (1983). "What some concepts might not be". *Cognition*, no. 13, pp. 263-308.

Ball, G. H. and D. J. Hall (1965). ISODATA, a novel method of data analysis and classification: Technical Report. Stanford, CA, Stanford University.

Barsalou, Lawrence W. (1983). "Ad hoc categories". *Memory & Cognition,* vol. 11, pp. 211-227.

Barsalou, Lawrence W. (1987). "The instability of graded structure: Implications for the nature of concepts". In U. Neisser (ed.), *Concepts and Conceptual Development: Ecological and Intellectual Factors in Categorization*. Cambridge: Cambridge University Press.

Berwick, Robert C. (1986). "Learning from positive-only examples". In Ryszard S. Michalski, Jaime G. Carbonell and Tom M. Mitchell (ed.), *Machine Learning,* vol. II. Los Altos, California: Morgan Kaufmann.

Bongard, Mikhail M. (1970). *Pattern Recognition*. New York: Spartan Books.

Brown, R. and C. Hanlon (1970). "Derivational Complexity and the Order of Acquisition in Child Speech". In J. R. Hayes (ed.), *Cognition and the Development of Language*. New Yowk: Wiley.

Buckley, Paul B. and C. B. Gillman (1974). "Comparisons of digits and dot patterns". *Journal of Experimental Psychology,* vol. 103, no. 6, pp. 1131-1136.

Chalmers, David (1996). *The Conscious Mind*. Oxford: Oxford University Press.

Crick, Francis (1994). *The Astonishing Hypothesis: The Scientific Search for the Soul*. New York: Simon & Schuster.

Dawkins, Richard (1996). *The Blind Watchmaker*. New York: Norton.

Dehaene, Stanislas (1997). *The Number Sense*. New York: Oxford University Press.

Dehaene, Stanislas, Ghislaine Dehaene-Lambertz*, et al.* (1998). "Abstract representations of numbers in the animal and human brain". *Trends in Neuroscience,* vol. 21, pp. 355-361.

Dehaene, Stanislas, Véronique Izard*, et al.* (2006). "Core Knowledge of Geometry in an Amazonian Indigene Group". *Science,* vol. 311, no. 20 January 2006, pp. 381-384.

Dennett, Daniel C. (1991). *Consciousness Explained*. Boston: Little, Brown and Company.

Diogenes, Laertius (1992). *Lives of Eminent Philosophers, I, Books 1-5*: Loeb Classical Library. Harvard.

Dreyfus, Hubert L. (1972). *What Computers Can't Do*. New York: Harper and Row.

Dreyfus, Hubert L. (1992). *What Computers Still Can't Do: A Critique of Artificial Reason*. Cambridge, Massachusetts: MIT Press.

Evans, Thomas G. (1968). "A program for the solution of a class of geometric-analogy intelligence-test questions". In M. Minsky (ed.), *Semantic Information Processing,* pp. 271-353. Cambridge, MA: MIT Press.

Fechner, G. T. (1860). *Elemente der Psychophysik*. Leipzig: Breitkopf & Härtel.

Foundalis, Harry E. (1999). "44 New Bongard Problems": http://www.cs.indiana.edu/~hfoundal/res/bps/bpidx.htm

Foundalis, Harry E. (2001). "Phaeaco, and the Bongard Problems": http://www.cs.indiana.edu/~hfoundal/research.html

Frede, Michael (1981). "Categories in Aristotle". In M. Frede (ed.), *Essays in Ancient Philosophy,* pp. 29-48: University of Minnesota Press, 1987.

Gentner, Dedre (1983). "Structure-mapping: a theoretical framework". *Cognitive Science,* vol. 7, pp. 155-170.

Goldstone, R. L. (1993). "Feature Distribution and Biased Estimation of Visual Displays". *Journal of Experimental Psychology: Human Perception and Performance,* vol. 19, no. 3, pp. 564-579.

Goldstone, R. L. and A. Kersten (2003). "Concepts and Categorization". In A. F. Healy and R. W. Proctor (ed.), *Comprehensive Handbook of Psychology,* vol. 4: Experimental psychology pp. 599-621. New Jersey: John Wiley and Sons, Inc.

Hampton, James A. (1979). "Polymorphous concepts in semantic memory". *Journal of Verbal Learning and Verbal Behavior*, no. 18, pp. 441-461.

Hampton, James A. (1982). "A demonstration of intransitivity in natural categories". *Cognition*, no. 12, pp. 151-164.

Hartigan, J. A. (1975). *Clustering Algorithms*. New York, NY: John Wiley and Sons, Inc.

Hofstadter, Douglas, R. (1977). "56 New Bongard Problems": http://www.cs.indiana.edu/~hfoundal/res/bps/bpidx.htm

Hofstadter, Douglas, R. (1979). *Gödel, Escher, Bach: an Eternal Golden Braid*. New York: Basic Books.

Hofstadter, Douglas R. (1985). *Metamagical Themas: Questing for the Essence of Mind and Pattern*. New York: Basic Books.

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

Hofstadter, Douglas R. (1995b). "A Review of Mental Leaps: Analogy in Creative Thought". *AI Magazine,* Fall 1995.

Hofstadter, Douglas R. (2001). "Epilogue: Analogy as the Core of Cognition". In Dedre Gentner, Keith J. Holyoak and Boicho N. Kokinov (ed.), *The Analogical Mind: Perspectives from Cognitive Science*. Cambridge, MA: MIT Press/Bradford Book.

Hofstadter, Douglas R. and Daniel C. Dennett (1981). *The Mind's Eye: Fantasies and Reflections on Self and Soul*. New York: Basic Books.

Holyoak, Keith J. and Paul Thagard (1995). *Mental Leaps*. Cambridge, Massachusetts; London, England: The MIT Press; a Bradford Book.

Hsu, Feng-Hsiung (2002). *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton, New Jersey: Princeton University Press.

Huffman, David A. (1971). "Impossible objects as nonsense sentences". In B. Meltzer and D. Michie (ed.), *Machine Intelligence 6,* pp. 295-324. Edinburgh, Scotland: Edinburgh University Press.

Jackson, Frank (1982). "Epiphenomenal Qualia". *Philosophical Quarterly,* vol. 32, pp. 127-136.

Jain, A. K., M. N. Murty*, et al.* (1999). "Data Clustering: a Review". *ACM Computing Surveys,* vol. 31, no. 3.

Johnson, Mark (1987). *The Body in the Mind: the Bodily Basis of Meaning, Imagination, and Reason*. Chicago: University of Chicago.

Kimberling, Clark (1998). *Triangle Centers and Central Triangles*. Winnipeg, Canada: Utilitas Mathematica Publishing Incorporated.

Kruschke, John K. (1992). "ALCOVE: An exemplar-based connectionist model of category learning". *Psychological Review*, no. 99, pp. 22-44.

Lakoff, George and Mark Johnson (1980). *Metaphors we Live by*. Chicago: University of Chicago.

Lakoff, George and Rafael E. Núñez (2000). *Where Mathematics Comes from: How the Embodied Mind Brings Mathematics into Being*. New York: Basic Books.

Lamberts, K. (1995). "Categorization under time pressure". *Journal of Experimental Psychology: General*, no. 124, pp. 161-180.

Linhares, Alexandre (2000). "A glimpse at the metaphysics of Bongard problems". *Artificial Intelligence,* vol. 121, no. 1–2, pp. 251–270.

Linhares, Alexandre (2005). "An active symbols theory of chess intuition". *Minds and machines,* vol. 15, pp. 131-181.

Lu, S. Y. and K. S. Fu (1978). "A sentence-to-sentence clustering procedure for pattern analysis". *IEEE Transactions, Syst. Man Cybern.*, no. 8, pp. 381-389.

Luria, Aleksandr, R. (1968). *The Mind of a Mnemonist: a Little Book about a Vast Memory*. (Translated from the Russian by Lynn Solotaroff.) Cambridge, Massachusetts: Harvard University Press.

Luria, Aleksandr, R. (1976). *Cognitive Development: Its Cultural and Social Foundations*. Cambridge, Massachusetts: Harvard University Press.

Maksimov, V. V. (1975). "Sistema, obuchayushtchayasya klassifikatsii geometricheskikh izobrazhenyi (A system for teaching the classification of geometric patterns)". In M. S. Smirnov and V. V. Maksimov (ed.), *Modyelirovaniye Obucheniya i Povyedyeniya (Modeling of Learning and Behavior, in Russian),* pp. 29-120. Moscow: Nauka.

Markman, Arthur B. (1999). *Knowledge Representation*. Mahwah, NJ: Erlbaum.

Marshall, James B. (1999). "Metacat: A Self-Watching Cognitive Architecture for Analogy-Making and High-Level Perception". Computer Science and Cognitive Science, Indiana University.

McCloskey, M. E. and S. Glucksberg (1978). "Natural categories: Well defined or fuzzy sets?". *Memory & Cognition,* vol. 6, pp. 462-472.

McQueen, J. (1967). *"Some methods for classification and analysis of multivariate observations"*. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability,* pp. 281-297.

Mechner, Francis (1958). "Probability relations within response sequences under ratio reinforcement". *Journal of the Experimental Analysis of Behavior*, no. 1, pp. 109-121.

Medin, D. L. and M. M. Schaffer (1978). "Context theory of classification learning". *Psychological Review*, no. 85, pp. 207-238.

Mehler, Jacques and Tom G. Bever (1967). "Cognitive capacity of very young children". *Science*, no. 158, pp. 141-142.

Miller, G. A. and P. N. Johnson-Laird (1976). *Language and Perception*. Cambridge, MA: Harvard University Press.

Mitchell, Melanie (1990). "Copycat: a computer model of high-level perception and conceptual slippage in analogy-making". Computer and Communication Sciences, University of Michigan.
Mitchell, Melanie (1993). *Analogy-Making as Perception*: MIT Press.
Mitchell, Tom M. (1978). Version Spaces: An Approach to Concept Learning. Computer Science Report CS-78-711, Stanford University.
Mitchell, Tom M. (1997). *Machine Learning*. New York: McGraw-Hill.
Moyer, Robert S. and Richard H. Bayer (1976). "Mental Comparison and the Symbolic Distance Effect". *Cognitive Psychology,* vol. 8, pp. 228-246.
Moyer, Robert S. and Thomas K. Landauer (1967). "Time required for Judgements of Numerical Inequality". *Nature,* vol. 215, pp. 1519-1520.
Moyer, Robert S. and Thomas K. Landauer (1973). "Determinants of reaction time for digit inequality judgments". *Bulletin of Psychological Society,* vol. 1, no. 3, pp. 167-168.
Murphy, Gregory, L. (2002). *The Big Book of Concepts*. Cambridge, MA: MIT Press.
Murphy, Gregory, L. and H. H. Brownell (1985). "Category differentiation in object recognition: Typicality constraints on the basic category advantage". *Journal of Experimental Psychology: Learning, Memory, and Cognition*, no. 11, pp. 70-84.
Murphy, Gregory, L. and D. L. Medin (1985). "The role of theories in conceptual coherence". *Psychological Review*, no. 92, pp. 289-316.
Nosofsky, Robert M. (1984). "Choice, similarity, and the context theory of classification". *Journal of Experimental Psychology: Learning, Memory, and Cognition*, no. 10, pp. 104-114.
Nosofsky, Robert M. (1992). "Exemplars, prototypes, and similarity rules". In A. Healy, S. Kosslyn and R. Shiffrin (ed.), *From Learning Theory to Connectionist Theory: Essays in Honor of W. K. Estes,* vol. 1 pp. 149-168. Hillsdale, NJ: Erlbaum.
Nosofsky, Robert M. and T. J. Palmeri (1997). "An exemplar-based random walk model of speeded categorization". *Psychological Review*, no. 104, pp. 266-300.
Papadimitriou, Christos H. (1994). *Computational Complexity*. Reading, Massachusetts: Addison-Wesley.
Parkman, John M. (1971). "Temporal aspects of digit and letter inequality judgments". *Journal of Experimental Psychology,* vol. 91, no. 2, pp. 191-205.
Piaget, Jean (1952). *The Child's Conception of Number*. New York: Norton.

Piaget, Jean (1954). *The Construction of Reality in the Child*. New York: Basic Books.

Pinker, Steven (1997). *How the Mind Works*. New York: W. W. Norton & Company.

Plato (1992). *Thaeetetus. Sophist*: Loeb Classical Library. Harvard.

Platt, John R. and David M. Johnson (1971). "Localization of position within a homogeneous behavior chain: Effects of error contingencies". *Learning and Motivation*, no. 2, pp. 386-414.

Posner, Michael I. and Marcus E. Raichle (1994). *Images of Mind*. New York: Scientific American Library.

Press, William H., Brian P. Flannery*, et al.* (1986). *Numerical Recipes: The Art of Scientific Computing*. New York: Cambridge University Press.

Rehling, John A. (2001). "Letter Spirit (Part Two): Modeling Creativity in a Visual Domain". Dissertation Thesis, Computer Science and Cognitive Science, Indiana University.

Rips, Lance J., E. J. Shoben*, et al.* (1973). "Semantic distance and the verification of semantic relations". *Journal of Verbal Learning and Verbal Behavior*, no. 12, pp. 1-20.

Rosch, Eleanor (1973). "On the internal structure of perceptual and semantic categories". In T.E. Moore (ed.), *Cognitive Development and the Acquisition of Language*. New York: Academic Press.

Rosch, Eleanor (1975). "Cognitive representations of semantic categories". *Journal of Experimental Psychology: General*, no. 104, pp. 192-233.

Rosch, Eleanor (1977). "Human categorization". In N. Warren (ed.), *Advances in Cross-Cultural Psychology,* vol. 1 pp. 177-206. London: Academic Press.

Rosch, Eleanor (1978). "Principles of categorization". In E. Rosch and B.B. Lloyd (ed.), *Cognition and Categorization,* pp. 27-48. Hillsdale, NJ: Elrbaum.

Rosch, Eleanor and C. B. Mervis (1975). "Family resemblance: Studies in the internal structure of categories". *Cognitive Psychology*, no. 7, pp. 573-605.

Rumbaugh, D. M., S. Savage-Rumbaugh*, et al.* (1987). "Summation in the chimpanzee (Pan troglodytes)". *Journal of Experimental Psychology: Animal Behavior Processes*, no. 13, pp. 107-115.

Rumelhart, David E., Geoffrey E. Hinton*, et al.* (1986). "Learning representations by back-propagating errors". *Nature*, no. 323, pp. 533-536.

Rumelhart, David E. and A. Ortony (1977). "The representation of knowledge in memory". In R. C. Anderson, R. J. Spiro and W. E. Montague (ed.), *Schooling and the Acquisition of Knowledge*. Hillsdale, NJ: Erlbaum.

Saito, Kazumi and Ryohei Nakano (1993). *"A Concept Learning Algorithm with Adaptive Search"*. In *Proceedings of the Machine Intelligence 14 Workshop,* pp. 347–363, Oxford University Press.

Schank, Roger C. (1982). *Dynamic Memory: A Theory of Learning in Computers and People*. New York: Cambridge University Press.

Schank, Roger C. and R. P. Abelson (1977). *Scripts, plans, goals, and understanding*. Hillsdale, New Jersey: Lawrence Erlbaum Press.

Schank, Roger C. and David B. Leake (1990). "Creativity and Learning in a Case-Based Explainer". In Jaime Carbonell (ed.), *Machine Learning*. Cambridge, Massachusetts: MIT / Elsevier.

Searle, John R. (1980). "Minds, Brains, and Programs". *Behavioral and Brain Sciences,* vol. 3, no. 3, pp. 417-457.

Shepard, Roger N., Dan W. Kilpatric*, et al.* (1975). "The Internal Representation of Numbers". *Cognitive Psychology,* vol. 7, pp. 82-138.

Shuford, Emir, H. (1961). "Percentage estimation of proportion as a function of element type, exposure time, and task". *Journal of Experimental Psychology,* vol. 61, no. 5, pp. 430-436.

Smith, E. E. and D. L. Medin (1981). *Categories and Concepts*. Cambridge, MA: Harvard University Press.

Smith, E. E. and D. N. Osherson (1984). "Conceptual combination with prototype concepts". *Cognitive Science*, no. 8, pp. 337-361.

Starkey, Prentice and R. J. Cooper, Jr. (1980). "Perception of numbers by human infants". *Science*, no. 210, pp. 1033-1035.

Strauss, M. S. (1979). "Abstraction of prototypical information by adults and 10-month-old infants". *Journal of Experimental Psychology: Human Learning and Memory*, no. 5, pp. 618-632.

Thompson, Richard F. (1993). *The Brain: A Neuroscience Primer*. New York, NY: W. H. Freeman and Company.

Thurston, L. L. (1927). "Psychophysical analysis". *American Journal of Psychology*, no. 38, pp. 368-389.

Treisman, Anne, M. (1980). "A Feature-Integration Theory of Attention". *Cognitive Psychology,* vol. 12, no. 12, pp. 97-136.

Treisman, Anne, M. (1986). "Features and Objects in Visual Processing". *Scientific American,* Nov 86, pp. 114-125.

Tversky, A. (1977). "Features of similarity". *Psychological Review*, no. 84, pp. 327-352.

Tye, Michael (1986). "The Subjective Qualities of Experience". *Mind*, no. 95, pp. 1-17.

van Oeffelen, Michiel P. and Peter G. Vos (1982). "A probabilistic model for the discrimination of visual number". *Perception & Psychophysics,* vol. 32, no. 2, pp. 163-170.

Waltz, David (1975). "Understanding line drawings of scenes with shadows". In P.H. Winston (ed.), *The Psychology of Computer Vision*. New York: McGraw-Hill.

Weber, E. H. (1850). "Der Tastsinn und das Gemeingefühl". In R. Wagner (ed.), *Handwörterbuch der Physiologie,* vol. 3, part 2, pp. 481-588. Braunschweig, Germany: VIeweg.

Welford, A. T. (1960). "The measurement of sensory-motor performance: Survey and reappraisal of twelve years progress". *Ergonomics,* vol. 3, pp. 189-230.

Wexler, K. and P. Culicover (1980). *Formal Principles of Language Acquisition*. Cambridge, Massachusetts: MIT Press.

Winograd, Terry A. (1972). *Understanding Natural Language*. New York: Academic Press.

Winston, Patrick, H. (1975). "Learning structural descriptions from examples". In P.H. Winston (ed.), *The Psychology of Computer Vision*. New York: McGraw-Hill.

Woodruff, Guy and David Premack (1981). "Primative mathematical concepts in the chimpanzee: Proportionality and numerosity". *Nature*, no. 293, pp. 568-570.

Zeki, Semir (1993). "The Visual Image in Mind and Brain". In W. H. Freeman (ed.), *Mind and Brain*. New York: W. H. Freeman and Company.

Curriculum Vitae

Harry E. Foundalis

Place of birth: Edessa, Greece

Date of birth: April 14, 1962

Education

- Ph.D. in computer science & cognitive science, Indiana University, 2006.
- M.Sc. in computer science, University of Alabama at Birmingham, 1987.
- B.S. in mathematics, University of Crete, Herakleion, Crete, Greece, 1985.

Publications

- "Evolution of Gender in Indo-European Languages". In *Proceedings of the Twenty-fourth Annual Conference of the Cognitive Science Society.* Fairfax, Virginia, August 2002.
- "A Type Scheme for a Theorem-Proving Language". M.Sc. Thesis, computer science, University of Alabama at Birmingham, Birmingham, Alabama, August 1987.