CHAPTER 7

Parallel and Serial Processing

GORDON D. LOGAN

INTRODUCTION

I remember myself as a preschooler asking my mother whether a person could do two things at once. I remember where we stood in the hallway when I asked the question, and I remember her answer: “Sometimes you can and sometimes you can’t. It depends.” I remember not being very satisfied with that answer. Too many years later, I find myself an expert on the question. I earned my doctorate asking it, and I spent a good part of my career asking it. After all this experience, my expert answer is this: Sometimes you can process things in parallel, and sometimes you process things in series. It depends. My mother was right all along, and my expert answer is no more satisfying than hers. In my expert opinion, the problem lies in the question, not in the answer. The question of parallel versus serial processing can be answered meaningfully only in the context of other issues and other concepts—the things on which “it depends.”

A major difficulty in answering the question lies in knowing what the answer means. How could one tell if processing were parallel or serial? In many ways, serial processes behave like parallel ones. They are affected similarly by experimental manipulations. One can trace the logic from the assumption of parallel versus serial processing to prediction of reaction time (RT) and accuracy, and the predictions are often very similar. This makes it hard, if not impossible, to argue from the data back to the theory. There are two routes to the same end, and given the end point, one cannot tell which route was taken. When parallel and serial processing predict the same results, the results do not distinguish the theories. This is the problem of mimicry. I was also into mimicry at an early age—it was a good way to annoy my brother Jack—but those experiences are more relevant to abnormal than to experimental psychology.

My purpose in writing this chapter is to discuss the methods that people use to ask whether processing is parallel or serial. In the years since I first asked my mother the question, researchers have been asking Mother Nature the same thing. They learned a lot about the things on which “it depends.” Much of the progress involved understanding the mimicry problem and finding ways to solve it. At the same time, researchers investigating attention and memory found themselves having to ask questions about serial versus parallel processing. They, too, made a lot of progress, though their conclusions were usually more specific. My goal is to explain the ways in which people ask the question and the issues that they confront in doing so. This chapter is intended more as a guidebook to orient people to the issues than as a user’s manual to teach specific methods (for reviews of the various issues, see J. Miller, 1988; Townsend, 1990; Van Zandt & Townsend, 1993).
BASIC DEFINITION

Put most simply, the question of parallel versus serial processing is about simultaneity and precedence in processing. Imagine two processes, A and B. If A and B go on simultaneously, then processing is parallel. If A precedes B or B precedes A, then processing is serial. Parallel and serial processing are illustrated in the top two panels of Figure 7.1. The

---

**Figure 7.1** Varieties of parallel and serial processing.

NOTE: The boxes represent stages of processing, and the arrows represent information flow between the boxes. The letters on the boxes represent different processes. \( P \) represents perceptual processes, \( M \) represents motor processes, and \( A \) and \( B \) represent central processes. The left column represents single-task situations, and the right column represents dual-task situations. The top row represents parallel processes; \( A \) and \( B \) are simultaneous, and neither precedes the other. The second row represents serial processes. \( A \) and \( B \) are never simultaneous. \( A \) begins and ends before \( B \) begins. The third row represents cascaded processes. \( B \) begins before \( A \) finishes, and \( A \) begins to transmit information to \( B \) before \( A \) finishes. \( A \) and \( B \) are simultaneous, but \( A \) precedes \( B \). The bottom row represents overlapping processes. \( A \) and \( B \) begin at different times but run simultaneously for some period.
left column represents the processes underlying performance of a single task (i.e., producing a single response to a single stimulus), and the right column represents the processes underlying performance in a dual task (i.e., producing two responses to two stimuli). In the parallel processing examples, A and B begin at the same time and end at the same time. In the serial processing examples, A begins and ends before B.

The simple question of whether processing is parallel or serial has intrigued every generation of cognitive psychologists. In the 1950s Broadbent (1957, 1958) argued that sensory processes were parallel and that cognitive ("perceptual") processes were serial. In the 1960s Sternberg (1966, 1969) argued that short-term memory scanning was serial rather than parallel. In the 1970s Shiffrin and Schneider (1977) argued that automatic processes were parallel and that controlled processes were serial. In the 1980s Treisman and colleagues (Treisman & Gelade, 1980; Treisman & Schmidt, 1982) argued that feature search was parallel and that conjunction search was serial. In the 1990s Meyer and Kieras (1997) argued that dual tasks could be performed in parallel, contradicting conventional wisdom, which says that dual tasks are carried out strictly in series (e.g., Pashler & Johnston, 1989; Welford, 1952). Also in the 1990s Rickard (1997) and I (Logan, 1988) argued over the serial nature of memory retrieval (see also Delaney, Reder, Straszewski, & Ritter, 1998). Many of these issues are not resolved, and the arguments will continue throughout the new century.

The zeitgeist has changed considerably over the generations, and the relative plausibility of serial and parallel processing has changed with it. In the early years, when the idea that mind is computation first took hold, people took the serial nature of computation quite seriously. Computers were serial, and people took the mind-as-serial-computer analogy quite literally. They were more likely to think of processing as serial than parallel. In recent years, at the end of the "decade of the brain," people have been impressed with the massive parallel nature of the brain and seem more likely to think that processing is parallel because that seems more brain-like than does serial processing. How serial behavior can emerge from a parallel brain has become an important question once again (see Lashley, 1951). I presume that the brain and the way it implements thinking have not changed with the zeitgeist.

COMPLICATIONS

The simple question of whether processing is parallel or serial is seductive because it seems so easy to answer. One need only be able to detect process A and process B and measure the times at which they occur. This turns out to be harder than it seems. There is no direct way to observe the occurrence of mental processes. One must infer their existence from changes in behavior that result from experimental manipulations. Most investigations of serial and parallel processing focus on accuracy and RT in relatively simple tasks. Accuracy and RT are final outcome measures that reflect the combined effects of all processes that go into producing a response. Most often, the question of serial or parallel processing concerns only some of the processes that contribute to a response, and separating the interesting processes from the uninteresting ones makes the inference from behavior to theoretical processes more complicated. In order to have a theory of the processes of interest, one must have also some kind of theory of the other processes and of how they combine to perform the whole task. The other processes may interact with the process of interest, and clever experiments may have to be done to tease them apart. Even with the
cleverest experiment, the chain of inference from observation to conclusion grows more complex.

Researchers have responded to these complications in two ways. One, which might be called the general class approach, is to create general classes of theory by combining binary (or ternary) distinctions among theories and then deriving in-principle predictions that distinguish the classes. To use a familiar example, visual search may be either parallel or serial and either exhaustive or self-terminating. The factorial combination yields four general classes of models from which predictions can be derived and tested (and these will be considered in the next section). Perhaps the most important results from this approach concern mimicry, showing that some models make the same predictions as other models so that observation of the predicted effects cannot distinguish the models.

The alternative approach, which might be called the specific theory approach, is to propose theories that account for specific sets of experimental data (e.g., Bundesen, 1990; Cave & Wolfe, 1990; Humphreys & Müller, 1993; Logan, 1996; Meyer & Kieras, 1997; Wolfe, 1994). Creating these theories requires making decisions about the binary distinctions studied in the other approach, so the general theories may fit into some category in the general class approach. The focus is different, however. The theories are often interpreted as models of the computations that underlie performance in the tasks they address, and the focus is on the nature of the computation and the way it is executed rather than on general properties of the computation, such as serial versus parallel processing. The most important results from this approach may be an increased understanding of the computational problems that underlie cognition and the discovery of some ways to solve them.

This chapter is organized around these two approaches to the problem of complexity. The main topics are organized around the general class approach, introducing the complexities one by one. The different complexities highlight different empirical situations and, consequently, illustrate different specific theories. Thus, the specific theory approach is embedded in the general class approach.

FOUR BASIC DISTINCTIONS

The general class approach involves making broad distinctions between classes of models and combining distinctions factorially to produce subclasses of models that differ from each other in fundamental ways. Most of the work has focused on four binary distinctions that combine to produce 16 classes of theory: parallel versus serial processing, discrete versus continuous processing, limited versus unlimited capacity, and self-terminating versus exhaustive search. From the perspective of this chapter, parallel versus serial processing is the focal distinction, and the others are complications. I begin with discrete versus continuous processing because it is the most general complication.

Discrete versus Continuous Processing

The first broad distinction is between discrete and continuous processes. Discrete processes transmit the information that they produce in a single step at a discrete point in time. Discrete transmission implies that processes begin and end at discrete points in time; they begin when they receive input (a transmission from a logically precedent process), and they end when they give output (a transmission to a logically subsequent process). The idea of discrete processing has been with us since the beginning of experimental psychology (e.g., Donders,
and these two complexities. The general complexities exit here and, consequently, is embedded.

S

oses making use of mod-dr factorially to find, differ from the other and parallel if they are simultaneous. Most of the distinctions of theory: discrete versus uniting versus perspective of processing are contrary continuum general processing.

 preach discrete processes produce in a me. Discrete es begin and begin when a logically discrete process begins. Donders, 1868). It prevailed throughout history (e.g., Sternberg, 1969), and it prevails today (e.g., Meyer & Kieras, 1997; Pashler & Johnston, 1989). The assumption of discrete processing makes formal modeling easier. It allows the modeler to derive predictions that are clear and intuitively compelling. Consequently, it has been popular among theorists. Indeed, much of the formal work on serial versus parallel processing that was done in the general class approach assumes discrete processing. Discrete processes fit the simple definitions of parallel and serial processing nicely. If A and B are discrete processes that are part of the same task, then they are serial if one precedes the other and parallel if they are simultaneous. The most commonly used techniques for analyzing RT assume discrete processing, including Donders’ (1868) subtractive method, Sternberg’s (1969) additive factors method, and the various analyses derived from them (e.g., Fisher & Goldstein, 1983; Goldstein & Fisher, 1991; Pashler & Johnston, 1989; Schweickert, 1978; Schweickert & Townsend, 1989; Townsend & Schweickert, 1989).

Donders’ (1868) subtractive method assumes that RT is the sum of the durations of a series of stages that extend from stimulus to response. Different tasks require different stages and different numbers of stages. The subtractive method considers special cases in which two tasks differ in exactly one stage; the remaining stages are the same in both tasks. In these special cases, the duration of the extra stage can be estimated by subtracting RT for the simpler task from RT for the more complex task. Sternberg’s (1969) additive factors method also assumes that RT is the sum of the durations of a series of stages, but the goal is to identify processing stages rather than to estimate their durations. Stages are identified with experimental variables that affect them. Variables that affect different stages will have additive effects because the durations of different stages add together to produce RT. Variables that affect the same stage will interact in a superadditive manner. Both methods assume discrete stages.

The alternative continuous processes transmit the information they produce gradually in an infinite number of infinitesimally small steps. Their beginning points and end points are not so clear, nor is the point at which they begin transmitting information to logically subsequent processes. It is clear, however, that logically subsequent processes can begin well before logically prior processes end. Continuous processes constantly report their current state to subsequent processes. Small changes in the current state propagate rapidly to the next stage and begin to affect its processing well before either stage has accumulated enough change to finish processing. Precedent processes are active simultaneously. Thus, continuous processes do not fit nicely into the simple definition of parallel and serial processes. The logical and temporal precedence suggest that processing is serial, but the simultaneous processing suggests that processing is parallel. To escape this quandary, some researchers refer to processes like this as cascaded, which reflects the mixture of precedence and simultaneity (e.g., McClelland, 1979). Other researchers think of continuous processes as parallel. Cascaded processes are illustrated in the third row of Figure 7.1. Single-task processing is on the left, and dual task processing is on the right.

Processes that are not continuous may be precedent and simultaneous if the precedence is only temporal and not logical (i.e., if the stage that begins second does not require information from the stage that begins first). In these cases, the processes may be discrete and parallel. Consider a single-task situation in which A and B are discrete parallel processes preceded by processes P’ and P’’,
respectively. If $P''$ takes longer than $P'$, A will start before B. At some later point, they both operate simultaneously. This is illustrated in the left side of the bottom row in Figure 7.1. In this case, the simultaneity and precedence of A and B do not require one to assume that they are continuous stages.

In dual-task situations, process A may be part of one task, and process B may be part of the other. Process A may begin before B, and thus be precedent, but A may end after B begins, and thus be simultaneous. In this case, the precedence is due to stimulus conditions (e.g., stimulus onset asynchrony) or to differences in the durations of processes prior to A and B. There is no logical contingency between A and B because they are parts of different tasks. Processes A and B are parallel and could be discrete. Their simultaneity and precedence do not require one to assume that they are continuous. Overlapping dual-task processes are illustrated in the bottom-right row of Figure 7.1.

A second case of precedent but simultaneous processing can occur in continuous tasks, such as typing, in which there is a chain of precedent processes and each process is active all the time. The discrete processing assumption can be salvaged if the different processes operate on different parts of the input. In typing "red ball," for example, perceptual processes may be working on "ball" while motor processes are busy with "red" (see, e.g., Butsch, 1932; Inhoff, Briihl, Bohemier, & Wang, 1992). An individual input would still be processed discretely, activating only one process at a time and jumping from one process to the next in a single discrete step. Jolicoeur, Tombu, Oriet, and Stevanovsky (in press) call this sort of processing pipelining. Pipelining speeds performance by allowing the system as a whole to process several inputs concurrently (in parallel) while each component of the system processes its input discretely (in series). For example, a three-stage model could process three inputs concurrently if each stage took the next input as soon as it was finished with the current one. By analogy, it takes four hours to build a single car in an assembly line, but the different stations on the line work on different cars at the same time, so the lag between successive cars is very short. The four-hours-per-car pipeline produces several cars in a single hour. Pipelining may save the discrete stage assumption, but it does not require it. Continuous processes may also be pipelined. It is interesting that the major formal theory of typewriting assumes continuous processing (Rumelhart & Norman, 1982).

Continuous processing has had a much shorter history than has discrete processing. It was proposed first around 1980 (e.g., C. W. Eriksen & Schultz, 1979; McClelland, 1979) as an alternative to discrete stage analyses of RT. Shortly afterward, the connectionist revolution began and adopted continuous processing as a fundamental assumption. Continuous processing is the "parallel" part of "parallel distributed processing" (e.g., McClelland, Rumelhart, & the PDP Research Group, 1986). Many connectionist models address response probability rather than RT and so do not address the issue of parallel versus serial processing in the usual sense. Connectionist approaches to RT are often very complicated and require simulation instead of mathematical analysis, and that makes it hard to produce general predictions (but see McClelland, 1993).

As J. Miller (1988, 1993) pointed out, discrete and continuous processes are at opposite ends of a continuum. Miller argued that information passes from one stage to another in chunks that can vary in size. The continuum that links discrete and continuous processes is defined by the chunk-size variable, which Miller called grain size. Grain size is determined by the number of chunks that must accumulate before processing terminates.
Discrete processes have the largest grain; processing terminates when one chunk is produced. Continuous processes have the smallest grain; processing terminates when an infinite number of infinitesimally small chunks are produced. Processes in the middle of the continuum have intermediate grain. Several chunks must be accumulated before processing terminates.

The issue of discrete versus continuous processing was a central focus of the empirical literature in the 1980s and 1990s, and the bulk of the evidence appears to contradict strictly discrete processes. Behavioral experiments by J. Miller showed evidence of continuous processing (e.g., 1982a, 1983, 1987). Meyer, Irwin, Osman, and Kounios (1988) found evidence of partial information with a response signal method that required subjects to respond on signal even if they had not finished processing. Psychophysiological experiments by Coles and colleagues showed evidence of concurrent, subthreshold activation of competing responses in electromyographic (EMG; Coles, Gratton, Bashore, Eriksen, & Donchin, 1985) and electroencephalographic (EEG; Gratton, Coles, Sirevaag, Eriksen, & Donchin, 1988) data in the B. A. Eriksen and Eriksen (1974) flanker task. J. Miller and Hackley (1992) and Osman, Bashore, Coles, and Donchin (1992) showed evidence of subthreshold activation of responses on go/no-go trials in go/no-go tasks. These data may rule out pure discrete processes, in which one chunk is enough to terminate processing, but they do not distinguish between continuous and intermediate grain-size partial-information discrete processes (see J. Miller, 1988; see also Meyer et al., 1988, vs. Ratcliff, 1988).

**Limited versus Unlimited Capacity**

The idea that the capacity for processing information is limited has been an essential part of cognitive psychology since the 1950s, particularly in research on attention (e.g., Broadbent, 1958) and memory (e.g., G. A. Miller, 1956). The idea that capacity may not always be limited has been a part of cognitive psychology for just as long (e.g., Sperling, 1960). Many careers have been made in deciding whether particular processes are limited or unlimited in capacity. The capacity issue intersects the parallel versus serial processing issue because limited capacity processes are often thought of as serial whereas unlimited capacity processes are often thought of as parallel (e.g., Treisman & Gelade, 1980; Van der Heijden, 1992). Parallel processes need not be unlimited in capacity. Indeed, _resource_ or _general capacity_ theories often assume parallel allocation of a limited pool of “mental energy” (e.g., Kahneman, 1973; Navon & Gopher, 1979; Norman & Bobrow, 1975), so processing is parallel but limited in capacity.

In the modern attention literature, the idea that serial processing is limited in capacity and parallel processing is unlimited in capacity plays out in two lines of investigation. One is the visual search literature that distinguishes between _preattentive processes_ that are parallel and unlimited in capacity and _focal attentive processes_ that are serial and limited in capacity (Cave & Wolfe, 1990; Duncan & Humphreys, 1989; Humphreys & Müller, 1993; Treisman & Gelade, 1980; Wolfe, 1994). The other is the memory and skill acquisition literature that distinguishes between _automatic processing_ that is parallel and unlimited in capacity and _controlled, strategic, effortful, or attentional processing_ that is serial and limited in capacity (Jacoby, 1991; Logan, 1988; Shiffrin & Schneider, 1977).

Processing capacity can be defined as the rate at which information is processed, expressed in units of information per unit time (Townsend & Ashby, 1983; Wenger & Townsend, 2000). From this perspective, capacity limitations are defined in terms of
changes in the processing rate for a particular element when another element is added to the task. Capacity is \textit{unlimited} if the processing rate does not change when another element is added to the task. That is,

\begin{equation}
  v(x, i)_N = v(x, i)_{N-1}
\end{equation}

where \( v(x, i)_N \) is the rate at which object \( x \) is compared to category \( i \) when there are \( N \) elements in the task, and \( v(x, i)_{N-1} \) is the rate at which object \( x \) is compared to category \( i \) when there are \( N - 1 \) elements in the task. Capacity is \textit{limited} if the processing rate slows down when another element is added to the task. That is,

\begin{equation}
  v(x, i)_N < v(x, i)_{N-1}.
\end{equation}

Capacity is \textit{fixed} if, when another element is added, the processing rate decreases in a manner in which the sum of the processing rates over all elements in the task remains constant. If capacity is allocated equally to all \( N \) elements, then

\begin{equation}
  v(x, i)_N = v(x, i)_{N-1} \frac{N - 1}{N}.
\end{equation}

If capacity is fixed and capacity allocation is not equal, then there is little constraint on a particular processing rate. The sum of rates is constrained to add to a constant \( C \), but the amount allocated to a particular process can vary between 0 and \( C \). The rate of processing for a particular process may even increase (e.g., if the person shifted from dividing attention among elements to focusing primarily on one element). Limited and fixed capacity are very hard to distinguish from each other.

\textbf{Capacity and Resources}

The idea of capacity is often confused with the idea of processing resources. Sometimes researchers use the terms interchangeably. In my view, however, “capacity” and “resources” have distinctly different meanings and one does not necessarily imply the other. The term “capacity” is relatively neutral theoretically; capacity is simply a rate measure, the amount of information processed per unit time. The term “resource” embeds the idea of capacity in complex theories of attention and performance that make many more assumptions than the simple assertion that performance can be measured in terms of processing rate (e.g., Kahneman, 1973; Navon & Gopher, 1979; Norman & Bobrow, 1975). Resource theories assume that capacity is fixed or severely limited, that capacity is a kind of mental energy that can be allocated selectively to activate mental processes, that capacity can be allocated in parallel, and that performance changes smoothly as capacity is added and withdrawn (Logan, 1997; Navon, 1984). Each of these additional assumptions is controversial, and not one of them is implied by the idea of capacity as a measure of processing rate. Resource theory may imply limited or fixed capacity, but limited or fixed capacity does not imply resource theory. Researchers should only say “resource” if they mean it. They should not say “resource” when they mean “capacity.”

\textbf{Capacity Limitations and Load Effects}

Many investigations of search tasks and dual tasks manipulate processing load. In search tasks, load depends on the number of items to be processed (i.e., the number of items in a search display, the number of items in a set of targets to be compared with the display, or both). In dual tasks, load depends on the difficulty of one or both tasks. Many people interpret load effects as evidence for capacity limitation. However, load effects can occur for several reasons other than capacity limitations (see, e.g., Duncan, 1980; Navon, 1984). The occurrence of load effects depends in part on the assumptions one makes about the cognitive architecture in which processing occurs. Load effects occur regardless of capacity limitations in certain \textit{independent race models}.
(e.g., Bundesen, 1990) and in equivalent Luce choice models (e.g., Luce, 1963). Consider an independent race model in which \( N \) objects in the display race to be categorized as members of category \( i \). If the distributions of finishing times are exponential in form, then the probability that object \( x \) wins the race is given by

\[
P("x \ is \ i") = \frac{v(x, i)}{\sum_{z=1}^{N} v(z, i)}.
\]

Marley and Colonius (1992) and Bundesen (1993) showed that independent race models such as this one are equivalent to Luce choice models in the sense that one can construct an independent race model that mimics the choice probabilities of any given Luce choice model. Consequently, Equation (4) describes response probabilities in Luce choice models as well as in independent race models.

Now consider what happens when another item is added to the display, so that \( N \) increases by 1. If processing capacity is fixed, \( P("x \ is \ i") \) will decrease because \( v(x, i) \) in the numerator of Equation (4) must decrease so that the sum of processing rates over the display—that is, \( \sum v(z, i) \) in the denominator of Equation (1)—remains constant. If processing capacity is limited but not fixed, \( P("x \ is \ i") \) will also decrease because \( v(x, i) \) decreases in the numerator and because the processing rate for the \( N \)th item, \( v(N, i) \), is added to the denominator, and the denominator increases. If processing capacity is unlimited, \( v(x, i) \) will remain the same in the numerator, but \( P("x \ is \ i") \) will decrease because the processing rate for the \( N \)th item, \( v(N, i) \), will be added to the denominator. Thus, for independent exponentially distributed race models and Luce choice models, load affects response probability whether capacity is fixed, limited, or unlimited. Therefore, contrary to popular opinion, the observation of load effects does not indicate fixed or limited capacity (see also Duncan, 1980; Navon, 1984).

### Functional and Stochastic Independence

Fixed or limited capacity suggests a kind of dependence among concurrent processes in that the rate of processing one element depends on the number of concurrently processed elements. However, formal models of fixed and limited capacity processes often assume independence. These ideas may seem contradictory, but they are not. They reflect different kinds of independence: functional independence and stochastic independence, respectively. Processes \( A \) and \( B \) are **stochastically independent** if the probability that \( A \) and \( B \) occur together is the product of the probabilities that each occurs separately. That is,

\[
P(A \cap B) = P(A) P(B).
\]

One tests stochastic independence by manipulating \( P(A) \) and \( P(B) \) and observing changes in \( P(A \cap B) \). If it remains predictable through the relationship in Equation (5), then \( A \) and \( B \) are stochastically independent. If it departs significantly from the relationship in Equation (5), then \( A \) and \( B \) are not stochastically independent. Stochastic independence is a very important assumption in mathematical modeling of parallel and serial processing. It simplifies the mathematics tremendously (see, e.g., Townsend & Ashby, 1983).

Processes \( A \) and \( B \) are **functionally independent** if the probability that \( A \) occurs is not correlated with the probability that \( B \) occurs. One tests functional independence by manipulating \( P(A) \) and observing changes in \( P(B) \). If \( P(B) \) does not change when \( P(A) \) changes, then \( A \) and \( B \) are functionally independent. If \( P(B) \) changes when \( P(A) \) changes, then \( A \) and \( B \) are functionally dependent. Functional independence has been important in studies in cognitive psychology and neuropsychology that rely on the logic of dissociations (e.g., Kelley & Lindsay, 1996). A dissociation occurs when a factor affects two processes differently—when processes are
functionally independent or negatively correlated. Functional independence represents a single dissociation; negative correlation represents a double dissociation.

Researchers often confuse stochastic and functional independence even though they are quite distinct conceptually. The two kinds of independence are tested by manipulating the same probability—\( P(A) \)—but conclusions about them are based on different probabilities. Stochastic independence rests on changes in \( P(A \cap B) \) when \( P(A) \) is manipulated; functional independence rests on changes in \( P(B) \) when \( P(A) \) is manipulated. Relationships between \( P(A \cap B) \) and \( P(A) \) are separate from relationships between \( P(B) \) and \( P(A) \), so the two kinds of independence address different aspects of the data. In particular, the functional dependence seen in fixed capacity and limited capacity models does not imply stochastic dependence. In a fixed or limited capacity system, taking capacity from \( A \) and giving it to \( B \) would increase \( P(A) \) and decrease \( P(B) \), signaling a violation of functional independence. However, stochastic independence rests on what happens to \( P(A \cap B) \), not to \( P(B) \). If \( P(A \cap B) \) changes in accord with the relationship in Equation (5), then \( A \) and \( B \) are stochastically independent even though they are functionally dependent.

**Capacity Limitations in Search**

Capacity limitations were central issues in the memory search literature of the 1970s and the visual search literature of the 1980s, where they were bound together with the issue of parallel versus serial processing. Sternberg (1966) contrasted serial processing with parallel processing in his classic paper on memory search, arguing that serial processing predicted the observed linear increase in RT with the number of items in the memory set to which the probe was compared (memory set size, or \( N \)), whereas parallel processing predicted a negatively accelerated increase (see Figure 7.2). The predictions for serial processing are clear: There is one comparison for each item in the memory set, and the mean time for successive comparisons is constant, so RT increases linearly with \( N \). Sternberg modeled parallel processing by assuming that memory search involves \( N \) independent parallel comparisons between the probe and the memory set. The probe is compared against all the items in the memory set before a decision is made (i.e., processing is exhaustive, as discussed later), so RT is the maximum of \( N \) independent samples from the distribution of comparison times. The maximum of \( N \) independent samples increases as a negatively accelerated function of \( N \) (see Gumbel, 1958), so RT should increase in that fashion if processing is parallel. The data contradicted that prediction, so Sternberg rejected parallel models.

Townsend and Ashby (1983) presented a derivation of Sternberg’s (1966) prediction with independently and identically distributed (i.i.d.) exponential comparison processes. There is one such process for each of the \( N \) items in the memory set, and the
comparisons continue until all have finished (i.e., processing is exhaustive). The time for
the last one to finish can be broken down into a sum of \textit{intercompletion times} that represents
the intervals between the finishing times of successive processes. The \( N \) processes begin
together but finish at different times, and
they can be ranked in the order in which they finish. Intercompletion time is the time be-
tween successive ranks. The processes race
against each other, each at the same rate, \( v \).
The first completion occurs when the fastest
of the \( N \) processes finishes. The distribution of
finishing times for the fastest runner in a race
between exponential distributions is an ex-
ponential distribution itself with a rate parameter
equal to the sum of the rate parameters of all
the runners in the race.\(^1\) Because there are
\( N \) runners with the same rate parameter, the
rate parameter for the first comparison to fin-
ish is \( Nv \), and the mean finishing time for the
first comparison is \( 1/Nv \). Because the
Distributions are exponential, the interval between
the first completion and the second is also ex-
ponentially distributed. This intercompletion
time can be thought of as a race between the
\( N - 1 \) remaining comparisons. The winner of
that race is exponentially distributed with a
rate parameter of \( (N - 1)v \) and a mean fin-
ishing time of \( 1/(N - 1)v \). The interval be-
tween the second and third comparison is also
exponentially distributed with a rate para-
ter of \( (N - 2)v \) and a mean finishing time of
\( 1/(N - 2)v \), and so on. Continuing this
process, mean finishing time for all \( N \) com-
parisons is

\[
E(T) = \frac{1}{Nv} + \frac{1}{(N-1)v} + \frac{1}{(N-2)v} + \cdots + \frac{1}{v} \\
= \frac{1}{v} \sum_{i=1}^{N} \frac{1}{i}.
\]

(6)

It is instructive to reverse the series to see
what happens as items are added to the memory
set:

\[
E(T) = \frac{1}{v} + \frac{1}{2v} + \frac{1}{3v} + \cdots + \frac{1}{Nv}.
\]

Each successive item that is added to the memory
set increases comparison time, but the
amount by which it increases gets progressively
smaller as \( N \) increases. This produces
negative acceleration in the function relating
mean RT to set size. This effect can be seen in
the predicted mean RTs from the i.i.d. ex-
ponential parallel (exhaustive) model plotted in
Figure 7.2.\(^1\)

Atkinson, Holmgren, and Juola (1969) and
Townsend (1974) noticed that Sternberg’s
(1966) parallel model assumed unlimited ca-
pacity. The rate at which individual compar-
isons were executed was the same for each
value of \( N \); for example, in Equation (6) it is
always \( v \). This assumption was central to the
derivation of the prediction (Gumbel, 1958),
so changing the assumption may change the
prediction. Atkinson et al. and Townsend dis-
covered that parallel processing could predict
the observed linear increase in RT with \( N 
if capacity was fixed and it could be reallo-
cated as soon as a comparison was finished
and the distribution of comparison times was
exponential. This was an important discovery

\(^1\) The probability density function for the minimum of two
samples’ probability density functions \( f(x) \) and \( g(x) \) is

\[
f(x) = f(x)[1 - G(x)] + g(x)[1 - F(x)]
\]

where \( F(x) \) and \( G(x) \) are cumulative distribution func-
tions (Townsend & Ashby, 1983). If \( f(x) \) and \( g(x) \) are
both exponential with rate parameters \( u_1 \) and \( u_2 \), respec-
tively, then the distribution of the minima of two samples
drawn from them is

\[
f(x) = u_1 \exp[-u_1 x] \exp[-u_2 x] + u_2 \exp[-u_2 x] \exp[-u_1 x] \\
= (u_1 + u_2) \exp[-(u_1 + u_2) x]
\]

which is an exponential distribution itself with a rate para-
ter that is the sum of the rate parameters for the two
runners in the race. This derivation can be generalized by
recursion to a race between \( N \) exponential distributions.
If the rate parameters for the different runners are all the
same, then the expected finishing time for a race between
\( N \) processes is \( 1/Nv \).
because it was one of the first formal demonstrations of mimicry.

The argument is similar to the argument for unlimited capacity processing. The time that the last comparison finishes can be broken down into a sum of the first finishing time and \( N - 1 \) intercompletion times. If capacity is fixed at \( C \) and allocated equally among all simultaneous comparisons during the first period before any of the comparisons finish, then the rate of processing for each individual comparison is \( C/N \). When the first comparison finishes, capacity is immediately reallocated, and the rate for the first intercompletion time is \( C/(N - 1) \). The rate for the second intercompletion time is \( C/(N - 2) \), and so on. The expected finishing time for all \( N \) processes can be computed by substituting these processing rates for the vs in Equation (6):

\[
E(T) = \frac{N}{NC} + \frac{N-1}{(N-1)C} + \frac{N-2}{(N-2)C} + \cdots + \frac{1}{C} \\
= \frac{1}{C} + \frac{1}{C} + \frac{1}{C} + \cdots + \frac{1}{C} \\
= \frac{N}{C}.
\]

Equation (7) shows that the mean finishing time for a parallel fixed-capacity exponential process with immediate reallocation increases linearly with set size (with a slope of \( 1/C \)), just as mean finishing time increases in serial models. Predicted RTs from the parallel fixed-capacity model are plotted along with the predictions of the parallel unlimited-capacity model in Figure 7.2.

The contrast between Equations (6) and (7) shows why fixed capacity produces a linear increase in RT with set size. When capacity is unlimited, as in Equation (6), each additional comparison takes progressively less time. The new comparison adds another runner to a race that is already fast; the more runners in the race, the smaller the new runner’s impact on the expected finishing time. When capacity is fixed and allocated equally among runners, adding a new runner reduces the amount of capacity that each runner gets, and the race slows down. The slowdown from the reduction in capacity per item compensates for the statistical speedup that results from having more runners in the race, so each new runner adds about the same amount of time to the race (see Equation [7]).

\( ^2 \)The argument depends on the idea that the finishing time for all \( N \) parallel processes can be broken down into the sum of the first finishing time and \( N - 1 \) intercompletion times. The focus on intercompletion times suggests that the race begins anew with one less runner when each comparison finishes, and this idea often runs counter to people’s intuitions about parallel processing. The runners that continue to run after the first one finishes were supposed to have begun running at the same time as the first runner, and all that time spent running ought to count for something. It seems that the interval between the first runner and the second should be a lot shorter than the time it took for the first runner to finish. The counterintuitive idea that the race begins anew and takes the same time to run each time, on average, stems from the “memoryless” property of exponential distributions. Because of that property, the probability that an event occurs before time \( t_1 + t_2 \) given that it has not occurred before time \( t_1 \) is equal to the probability that the event occurs in the first \( t_2 \) time units. The relationship goes as follows:

\[
P(T < t_1 + t_2 | T > t_1) = \frac{P(T < t_1 + t_2 \cap T > t_1)}{P(T > t_1)}
\]

\[
= \frac{F(t_1 + t_2) - F(t_1)}{1 - F(t_1)}
\]

\[
= \frac{1 - \exp[-v(t_1 + t_2)] - (1 - \exp[-v(t_1)])}{1 - (1 - \exp[-v(t_1)])}
\]

\[
= \frac{\exp[-v(t_1)] - \exp[-v(t_1 + t_2)]}{\exp[-v(t_1)]} \\
= \frac{\exp[-v(t_1)] - \exp[-v(t_1)]\exp[-v(t_2)]}{\exp[-v(t_1)]} \\
= \frac{\exp[-v(t_1)](1 - \exp[-v(t_2)])}{\exp[-v(t_1)]} \\
= 1 - \exp[-v(t_2)] = F(t_2) = P(T < t_2).
\]

In other words, the distribution of finishing times for events in the race that continues after the first event finishes at time \( t_1 \) is the same as the distribution of finishing times for a race with the same number of runners that begins at time 0.
In my view, Townsend's (1974) demonstrations of mimicry between serial and parallel processes signaled the beginning of the end of a period of intense interest in memory search. Models that could account for the linear increase in mean RT with set size proliferated, and the empirical arena shifted to other aspects of the data, such as sequential effects and RT distributions, and mimicry issues appeared in these other aspects as well (for reviews, see Luce, 1986; Townsend & Ashby, 1983). By the early 1980s, research on parallel versus serial processing in memory search seemed to have reached a stalemate, and interest shifted elsewhere.

Around 1980, inspired by Treisman and Gelade's (1980) elegant experiments, attention researchers embraced the issue of parallel versus serial processing in visual search. Treisman and Gelade showed that search RT for simple targets such as a red item among green items or an X among Os (i.e., feature search) was independent of the number of items in the display (i.e., display size), whereas search RT for conjunctive targets such as a red X among red Os and green Xs (i.e., conjunction search) increased linearly with N. Their feature integration theory interpreted their data as indicating that feature search was parallel and that conjunction search was serial. Citing Townsend (1971), they acknowledged the possibility that the linear functions in conjunction search could be produced by parallel processes, but they preferred to interpret them as evidence for serial processing. The burgeoning literature on feature and conjunction search followed their lead, mostly ignoring the mimicry issue. The functions relating RT to display size were markedly different in feature search and conjunction search, and that difference was enough to sustain the idea that the tasks were performed by different processes, regardless of the possible mimicry.

Wolfe and colleagues proposed guided search theory as an improvement on feature integration theory (Cave & Wolfe, 1990; Wolfe, 1994; Wolfe, Cave, & Franzel, 1989) but still interpreted the linear functions in conjunction search as evidence for serial processing. Duncan and Humphreys (1989) were more neutral on the issue, interpreting the slopes of RT × display size functions as measures of search efficiency. Humphreys and Müller (1993) made the mimicry problem concrete by proposing search by recursive rejection that accounted for flat RT × display size functions in feature search and for steep, linear RT × display size functions in conjunction search with the same parallel model. Researchers pitting their model against feature integration theory or guided search theory must grapple with the issue of parallel versus serial processing.

Several researchers proposed compromise models that sample regions of the display in series but process items within regions in parallel (Duncan & Humphreys, 1989; Grossberg, Mingolla, & Ross, 1994; Logan, 1996; Treisman & Gormican, 1988). It seems to me that these models are on the right track if theories of visual search are to be generalized to real-world behavior. Although we do spend more and more time staring at computer screens like in visual search experiments, even the most sedentary among us spends a lot of time each day searching large-scale environments such as refrigerators, rooms, shopping malls, streets, and freeways. The gradient of retinal acuity forces us to move our eyes to search these large-scale environments, imposing serial processing on our search behavior. Search may be parallel within fixations, processing all items in the fovea and parafovea.
one task is strongly affected by the requirement to do another (for a review, see Pashler, 1994a). The contrast between serial and parallel processing has played out in this literature since the beginning of the modern era of cognitive psychology. The first modern theory of dual-task performance was Welford’s (1952) single channel theory, which assumed that people dealt with dual tasks in series. Welford’s idea was adopted and extended by Broadbent (1957, 1958), who made serial processing a core property of attention within and beyond dual-task situations.

Serial processing was the favored explanation of dual-task performance until the end of the 1960s, when resource theory arose (e.g., Kahneman, 1973; Moray, 1967; Posner & Boies, 1971). Resource theories argued that people perform dual tasks in parallel but with less efficiency than in single-task conditions because capacity is severely limited. Kahneman proposed the broadest theory. He applied a single-resource theory to all problems in attention but focused especially on dual-task performance. In his theory, resources were allocated in parallel whenever it was beneficial to do so. By the end of the 1970s, single-resource theory was replaced by multiple-resource theory (e.g., Navon & Gopher, 1979), but dual-task performance was still thought to be parallel. Multiple-resource theory agreed with single-resource theory in suggesting that a single resource could be allocated in parallel, but it went beyond single-resource theory in arguing that different resources could also be allocated in parallel. This added a new wrinkle: Two tasks that demanded different resources could go on in parallel without interference.

In the middle of the 1980s, Pashler (1984; Pashler & Johnston, 1989) resurrected single-channel theory and derived new predictions from it that confirmed the idea of serial processing in dual-task situations. Predictions derived from resource theory, on the hypothesis that dual-task processing is parallel, fared less well (e.g., Pashler, 1994b).

In the 1990s the parallel versus serial issues played in two areas, one empirical and one theoretical. The empirical arena contrasted dual-task effects seen in speeded tasks, such as the psychological refractory period (PRP) procedure championed by Welford (1952) and Pashler (1984), with effects seen in unspeeded tasks with brief exposures, such as the attentional blink procedure introduced by Raymond, Shapiro, and Arnell (1992) and Chun and Potter (1995). The speeded tasks seemed to tax a central bottleneck that selected one response at a time (i.e., serial processing), whereas the unspeeded tasks seemed to tax central resources involved in forming perceptual representations (i.e., parallel processing). In the theoretical arena, Meyer and Kieras (1997) challenged the fundamental idea underlying both central bottlenecks and resource theories of dual-task interference, arguing that dual-task effects were often artifacts of the strategies that subjects adopted to deal with dual-task experiments rather than central capacity limitations (see also Logan & Gordon, 2001). They focused primarily on the PRP situation, explaining PRP effects in a model that had no central bottlenecks or central capacity limitations, but their argument generalizes to many dual-task situations.

Researchers noted the potential for mimicry between serial and parallel explanations of dual-task interference early on. A serial process that alternated rapidly enough would seem like a parallel process. This idea was exploited in early multiuser operating systems for serial computers: If the computer switched back and forth between users rapidly enough, the users could think they were operating the computer at the same time. In the empirical arena, subjects could seem to be performing two tasks in parallel even though they were switching rapidly between them (see, e.g., Broadbent, 1982). This kind of mimicry
seems amenable to empirical testing. One can measure the time required to switch attention in order to see if it switches rapidly enough. Unfortunately, there is no consensus on methods for estimating the time required to shift attention, and estimated switching time varies across methods by two orders of magnitude. Estimated switching time is fastest in search tasks, where it may be on the order of 20 ms to 40 ms, and slowest in cuing tasks, where it may be on the order of 1,000 ms to 2,000 ms. Duncan, Ward, and Shapiro (1994; see also Ward, Duncan, & Shapiro, 1996) estimated the time required to switch attention in an attentional blink task and argued that it was too slow to support serial processing in search tasks. Moore, Egeth, Berglan, and Luck (1996) contested that conclusion, arguing that their procedure substantially overestimated switching time.

Recent studies of the PRP procedure have used the parallel versus serial issue to localize a hypothesized bottleneck in processing (e.g., Pashler, 1984; Pashler & Johnston, 1989). By hypothesis, stages prior to the bottleneck can go on in parallel within and between tasks, whereas the bottleneck stage is strictly serial. Task 1 and Task 2 can be processed in parallel up to the stage at which they require the bottleneck. At that point, one task gets the bottleneck (usually Task 1) and the other task has to wait for it (usually Task 2). The period during which Task 2 has to wait for the bottleneck is called slack, and the bottleneck can be located by finding the locus of the slack in the processing chain. Processes prior to the bottleneck are parallel and so can begin as soon as they receive input. There is no slack before them. The slack period appears just before the bottleneck begins, so localizing the slack also localizes the bottleneck.

The method, often called the locus of slack method, is illustrated in Figure 7.3. It involves a factorial experiment with at least two factors: the stimulus onset asynchrony (SOA) between the stimulus for Task 1 (S1) and the stimulus for Task 2 (S2) and a manipulation of Task 2 difficulty. SOA usually produces a strong main effect on RT to S2 (RT2), and the difficulty manipulation is chosen so that it also produces a strong main effect on RT2. The key datum is the interaction between SOA and the Task 2 difficulty variable. Task 2 difficulty variables that affect stages prior to the bottleneck will produce underadditive interactions with SOA, Task 2 difficulty variables that affect stages at or after the bottleneck will produce null or additive interactions with SOA (see Pashler & Johnston, 1989; see also Fisher & Goldstein, 1983; Goldstein & Fisher, 1991; Schweickert, 1978; Schweickert & Townsend, 1989; Townsend, 1984; Townsend & Schweickert, 1989).

Figure 7.3A shows why variables that affect prebottleneck stages produce underadditive interactions with SOA. The top part shows flow charts for Task 1 and easy and hard versions of Task 2 with a short SOA. Because SOA is short, Task 2 has to wait for the bottleneck stage, and there is slack in the easy version of Task 2. The hard version of Task 2 has time to catch up to the easy version during the slack period, and it is almost finished when the slack period ends. The effects of the Task 2 difficulty manipulation are absorbed into the slack, so the Task 2 difficulty manipulation has only a small effect on RT2. The bottom part shows the same Task 1 and Task 2 conditions when SOA is long and Task 2 does not have to wait for the bottleneck. There is no slack period to absorb the Task 2 difficulty effect, so it appears full-blown in RT2. When RT2 is plotted against SOA, as in the right side of Figure 7.3A, Task 2 difficulty effects are smaller when the SOA effects are larger. Consequently, Task 2 difficulty interacts underadditively with SOA.

Figure 7.3B shows why variables that affect bottleneck stages produce additive or null interactions with SOA. The top part shows
Figure 7.3 Postponing prebottleneck processes vs. postponing bottleneck and postbottleneck processes.

NOTE: Panel A: The effects of postponing prebottleneck processes. The left side presents flow diagrams of processes underlying Task 1 and Task 2. P represents prebottleneck perceptual processes, B represents bottleneck processes, and M represents motor processes. The top part represents short stimulus onset asynchrony (SOA). Stage P' is a prolonged version of stage P. The short SOA causes Task 2 to wait for the bottleneck stage B, and both P and P' have time to finish during the "slack" period while Task 2 waits for the bottleneck. The effect of P versus P' on RT2 is given by ΔRT and is plotted on the RT2 × SOA graph beside the flow diagram. The bottom part represents long SOA. Task 2 does not have to wait for the bottleneck, so the effect of P versus P' propagates to RT2. The ΔRT value is much larger and results in an underadditive interaction between SOA and P versus P' when plotted on the RT2 × SOA graph beside the flow diagram. Thus, prolonging prebottleneck Task 2 processes produces underadditive interactions between Task 2 difficulty variables and SOA.

Panel B: The effects of postponing bottleneck or postbottleneck processes. The left side presents flow diagrams of processes underlying Task 1 and Task 2. In this panel, the bottleneck stage B is prolonged in Task 2. The effects of prolongation appear undiminished in RT2 because the Task 2 bottleneck processing does not begin until Task 1 is finished with the bottleneck, regardless of SOA. The right side plots the effects in a graph of RT2 × SOA. The effect of B versus B' is clearly additive with SOA. Thus, prolonging bottleneck or postbottleneck Task 2 processes produces additive (null) interactions between Task 2 difficulty variables and SOA.
flow charts for Task 1 and easy and hard versions of Task 2, but now difficulty affects the bottleneck stage. Because SOA is short, Task 2 has to wait for the bottleneck. Because the Task 2 difficulty manipulation affects the bottleneck stage, which has to wait, the easy and hard versions start at the same time, and the difficulty effect appears full-blown in RT2. The bottom part shows the same conditions with a long SOA. Task 2 does not have to wait, and the difficulty manipulation appears full-blown in RT2 once again. Its magnitude is the same as in the short SOA condition, so the joint effects of SOA and Task 2 difficulty, plotted on the right side of the panel, are additive; the interaction is null.

The locus of slack logic is a generalization of Sternberg’s (1969) *additive factors method* for decomposing single tasks into component stages. The locus of slack logic is also a special case of a much broader and more formal generalization of the additive factors logic by Schweickert, Townsend, and Fisher (e.g., Fisher & Goldstein, 1983; Goldstein & Fisher, 1991; Schweickert, 1978; Schweickert & Townsend, 1989; Townsend, 1984; Townsend & Schweickert, 1989). In the general logic, underadditive interactions between difficulty variables are often diagnostic of parallel processes, whereas additive or null interactions are often diagnostic of serial processes (Townsend, 1984). These principles cannot be applied universally, however. The issue of parallel versus serial processing remains complicated; interested readers should refer to the original sources.

Like many other tests of parallel versus serial processing, the locus of slack method assumes that processing in the bottleneck stage is discrete, not continuous. Task 1 finishes with the bottleneck at a distinct point in time, and Task 2 starts using the bottleneck at another distinct point in time. The latter never precedes the former. However, some recent data from the PRP procedure suggest that response selection—a favorite candidate for bottleneck processing—may not be discrete. Hommel (1998) and Logan and Schulkind (2000) showed that RT to the first PRP stimulus was influenced by the response category of the second PRP stimulus, speeding up if that response category was congruent with its own and slowing down if it was incongruent. This suggests that Task 2 response selection began before Task 1 response selection finished, arguing against the hypothesis that response selection is discrete and serial.

The seriousness of the consequences of violating the assumption of discrete processing remains to be seen. Although it is clear that the logic of the locus of slack model and the generalizations of it were developed on the assumption of discrete processing, it is not clear whether a continuous model would make different predictions. As yet, no one has worked out the predictions, although several investigators are working on applications of a single-resource theory to the SOA × difficulty manipulation factorial experiments, and the theory appears to be able to predict the same kinds of underadditive and additive features as can a serial discrete model. That would be a most interesting result.

**Self-Terminating versus Exhaustive Search**

One of the most important complications of the issue of parallel versus serial processing in the search literature is how processing stops. Search tasks require several comparisons between items in the display and items in memory. At some point, the comparisons stop, and the results are passed on to the next stage so that, ultimately, they can be reported. The key question is, how do the comparison processes stop? Traditionally, there have been two alternatives: Search is self-terminating or exhaustive. Search is *self-terminating* if it stops as soon as a target is found or *exhaustive* if it
continues until all of the comparisons are finished, regardless of whether or when a target is found.

The issue of self-terminating versus exhaustive search interacts with the issue of parallel versus serial processing in search tasks. The way that search terminates determines the number of items that need to be compared (i.e., $N$ if search is exhaustive; less than $N$ if search is self-terminating), and parallel versus serial processes are distinguished in terms of the effects of the number of items in the display or in the memory set. If search is self-terminating, the number compared may not equal the number displayed or memorized. Indeed, the predictions of unlimited capacity parallel models and the mimicry of fixed capacity parallel models and serial models described earlier requires the assumption that search is exhaustive. If search is self-terminating, parallel unlimited capacity models can predict a null effect of set size (i.e., no increase in RT with set size). In an unlimited capacity model, the rate at which the target comparison is executed is the same regardless of the number of concurrent nontarget comparisons, so the time required to find the target should be independent of display size. (Display size is usually manipulated by varying the number of nontargets.)

Intuition suggests that search should always be self-terminating, because that seems most efficient. Perhaps the most remarkable aspect of Sternberg’s (1966) data is that they suggested that search is exhaustive. Sternberg noted that self-terminating search requires the system to decide whether to terminate search after each comparison, whereas exhaustive search requires only one decision after all the comparisons are finished. Sternberg argued that if the decision to terminate was costly, then exhaustive search may be more efficient than self-terminating search. The cost of extra comparisons that finish after the target has been found may be small compared to the accumulated cost of deciding whether to terminate search after each comparison, particularly if the number of items to be compared is small. Search may become self-terminating with larger numbers of items. Indeed, Sternberg studied memory sets of one to five items. Visual search experiments show evidence of exhaustive search when the number of items in the display (display size) varies between one and five (e.g., Atkinson et al., 1969) and evidence of self-terminating search when display size varies over a larger range (e.g., 4–40; Treisman & Gelade, 1980).

The issue of self-terminating versus exhaustive search focuses primarily on the interaction between display size or memory set size and target presence or absence. In general, self-terminating search predicts superadditive interactions between set size and target presence, whereas exhaustive search predicts additive or null interactions. Often, RT increases linearly with set size, and the predictions are expressed in terms of ratios of the slopes of the functions relating RT to set size. Self-terminating search is often said to predict that the ratio of the target-absent slope to the target-present slope is 2:1, whereas exhaustive search predicts a ratio of 1:1. Sternberg’s (1966) data showed the 1:1 ratio, so he rejected self-terminating search in favor of exhaustive search. Treisman and Gelade’s (1980) data showed the 2:1 ratio, so they rejected exhaustive search in favor of self-terminating search.

The predicted slope ratios follow from the expected number of comparisons when the target is present versus absent. With exhaustive search, subjects perform all comparisons whether the target is present or absent, so the expected number of comparisons for set size $N$ is $N$ for both target-present and target-absent trials. With self-terminating search, subjects perform all comparisons only if there is no target; self-terminating search is exhaustive on target-absent trials. Target-absent
ether to ter-
ison, partic-
e compared 
terminating
is. Indeed,
one to five
show evi-
tual termi-
number
size) varies
son et al.,
sing search
targer range
1980).
versus ex-
y on the in-
or memory
absence. In
predicts su-
s set size and
ive search
ions. Often,
ize, and the
ms of ratios
relating RT
ch is often
the target-
slope is 2:1,
as a ratio of
wed the 1:1
ating search
reisman and
2:1 ratio,
h in favor of
ow from the
is when the
ith exhaust-
mparisons
sent, so the
s for set size
and target-
ing search,
ly if there
earch is ex-
target-absent

trials require \( N \) comparisons if set size is \( N \). On target-present trials, however, processing can terminate whenever a target is found. If search is random, the target could be found after the first, second, third, or later comparison but on average would be found after \((N + 1)/2\) comparisons. For the same set size, self-terminating search requires about twice as many comparisons for target-absent trials as for target-present trials. If the slope of the function relating RT to the number of comparisons is the same for target-absent and target-present trials (i.e., if each comparison takes the same amount of time), then the slope of the function relating RT to set size will be twice as large for target-absent trials.

Van Zandt and Townsend (1993) showed that self-terminating search does not always predict a 2:1 slope ratio and that in some cases it predicts a 1:1 slope ratio (see also Townsend & Colonius, 1997). That suggests a potentially paralyzing mimicry. However, they showed that exhaustive models almost always predict a 1:1 slope ratio, so finding a ratio other than 1:1 allows us to reject exhaustive models in favor of self-terminating models.

The issue of self-terminating versus exhaustive search usually focuses on target-present trials, asking whether subjects stop when they find a target. Chun and Wolfe (1996) focused on target-absent trials and asked how subjects decide to stop searching when they do not find a target. Environments are usually cluttered with many things, but people ignore most of them when they search for something. When I search for my car in a parking lot, I look at the cars, not the trees and buildings and people. Chun and Wolfe argued that subjects set some criterion for similarity to the target object and restrict their search to items that are similar to the target. They argued that the criterion is set dynamically, decreasing if the distractors are dissimilar to the target and increasing if the distractors are similar to the target. This adjustment process can reduce the number of items examined on target-absent trials to a value that is substantially smaller than \( N \), and that may affect the ratio of target-absent to target-present slopes. In their view, self-terminating search does not necessarily predict a 2:1 slope ratio.

Data beyond mean RT can be used to distinguish between self-terminating and exhaustive search. Townsend and Ashby (1983) noted that the models make different predictions about the variance of RT. For serial exhaustive search, the variance of the comparison times is simply the sum of the variances in the processing times for each comparison. If the variances are all equal, then

\[
\text{Var}_{\text{exhaustive}} = N \text{Var}(T)
\]  

where \( \text{Var}(T) \) is the variance in a single comparison time. The same prediction can be derived for parallel exhaustive search, where \( T \) is intercompletion time (Townsend & Ashby, 1983).

Equation (8) also describes the relation between variance and set size for target-absent trials in serial self-terminating search. For target-present trials, however, serial self-terminating search predicts a stronger increase in variance with set size:

\[
\text{Var}_{\text{self-terminating}} = \frac{N + 1}{2} \text{Var}(T) + \frac{(N - 1)(N + 1)}{12} E(T)^2.
\]

The variance on target-present trials depends on the variance in time required for each comparison, as it did on target-absent trials, but it also depends on variation in the number of items compared before the target is found. This additional source of variation makes the overall variance increase faster with \( N \) on target-present than on target-absent trials. Again, similar arguments can be made for parallel self-terminating processes (see Townsend & Ashby, 1983).
ALTERNATIVES TO PARALLEL PROCESSING

The four basic distinctions that drove the general class approach were discovered in the 1950s and 1960s. Since then, two new alternatives to parallel processing have come on the scene, one in the attention literature that deals with the effects of redundant signals and one in the skill acquisition literature that addresses the development of automaticity.

Statistical Facilitation versus Coactivation

Parallel processing is an important issue in divided attention. When participants look (or listen) for a target in two channels (two display locations or two acoustic sources), they respond faster if a target appears in both channels than if it appears in only one of them (Miller, 1978, 1982b). This redundant signals effect is interesting because it rules out most of the parallel and serial models considered so far in this chapter. Serial and parallel exhaustive models can be ruled out by the effect itself. If targets and distractors take the same amount of time to process, they predict no advantage of redundant targets (but see Townsend & Nozawa, 1997). Parallel and serial self-terminating models can predict the occurrence of the effect—processing can stop as soon as one target is found, and that will be faster when there are two targets—but they can be ruled out in many cases because their quantitative predictions underestimate the observed effect.

Serial self-terminating models predict an advantage of redundant targets when each channel contains either a target or a distractor. If search is random and targets are assigned randomly to channels, then the first object examined will always be a target on redundant trials, but it will only be a target half of the time on single-target trials. On the other half of single-target trials, the distractor will be examined first, so RT will increase. Averaging the two kinds of single-target trials produces a mean RT that is slower than the mean RT for redundant target trials, thus predicting a redundant signals effect. The observed effects are often larger than these models predict, however (see Miller, 1982b). Moreover, serial self-terminating models predict no advantage when no distractors are presented (i.e., targets appear alone or in tandem), because the first object examined will always be a target, and redundant signals effects are often found under those circumstances (Miller, 1978, 1982b).

The strongest candidate among the models discussed so far is the class of independent unlimited-capacity parallel self-terminating models. They predict statistical facilitation with redundant signals. The time to find a target when two are present is the minimum of the times required to find each target alone, and the minimum is generally faster than the mean of the parent distributions from which it is sampled (Gumbel, 1958). This argument extends to distributions as well as means, and Miller (1978, 1982b) developed it into a general test for cumulative distribution functions.

The distribution of minima sampled from two parent distributions can be constructed from the parent distributions themselves. If the samples are independent, then

\[ P(\min(T_1, T_2) < t) = P(T_1 < t) + P(T_2 < t) - P(T_1 < t \cap T_2 < t) \]

(10)

where \( P(\min(T_1, T_2) < t) \) is the observed cumulative RT distribution with redundant signals and \( P(T_1 < t) \) and \( P(T_2 < t) \) are the observed cumulative distributions with targets in channels 1 and 2, respectively. The final term is not easy to observe directly, so Miller (1978, 1982b) suggested rearranging the equation to produce an inequality called the race model inequality that investigators
could use to test the predictions of independent unlimited-capacity parallel self-terminating models:

\[ P(\text{min}(T_1, T_2) > t) \leq P(T_1 > t) + P(T_2 > t). \]  

(11)

The race model inequality has the advantage over Equation (10) in that all the terms in it are observable, so it can be used to test empirical data.

Miller (1978, 1982b) and others tested the race model inequality in several data sets. Amazingly, the data violated the predicted inequality. Performance with redundant signals was better than what was predicted from the most efficient parallel model. In order to explain the redundant signals effect, something more than unlimited-capacity parallel processing had to be proposed. Miller (1978, 1982b) proposed coactivation, which he viewed as resulting from interactions and cross talk between concurrent channels. Mordkoff and Yantis (1991; see also Mordkoff & Egeth, 1993) suggested an interactive race model, which they simulated and applied to their data. Townsend and Nozawa (1997) suggested the idea of supercapacity, an alternative to fixed, limited, and unlimited capacity in which the processing rate actually increases as load increases.

**Races versus Mixtures**

Parallel processing is an important issue in skill acquisition. Logan’s (1988) instance theory of automaticity seems salient in this context. Instance theory explains automatization as a transition from a general algorithm that is used to solve novel problems and memory retrieval of past solutions to familiar problems. The theory assumes that people store memory traces, or instances, of each encounter with each stimulus, so a task-relevant knowledge base builds up with practice. The theory assumes that people retrieve memory traces when familiar stimuli are encountered and that retrieval is a self-terminating unlimited-capacity parallel process, also known as an independent race model. Instance theory explains the learning curve—the ubiquitous speedup in RT with practice—as statistical facilitation from a race between the instances in memory, whose number grows with each encounter with the stimulus.

Newell and Rosenbloom (1981) reviewed 50 years of research on skill acquisition and declared the power law; RT decreased as a power function of practice:

\[ RT = a + b N^{-c}. \]  

(12)

where \( a \) is an irreducible asymptote, \( b \) is the amount by which RT can change over learning, and \( c \) is the learning rate. Logan (1992) reviewed studies published in the 10 years after Newell and Rosenbloom’s paper and found power function learning in each of them. A typical power function with \( a = 500 \), \( b = 500 \), and \( c = 0.5 \) is plotted in Figure 7.4.

![Power vs. exponential learning curves](Figure 7.4)  

Mean reaction time (RT) as a function of the number of practice trials for a power function learning curve (solid line) and an exponential function learning curve (dotted line).

**NOTE:** The power function was generated from the equation \( RT = 500 + 500N^{-0.5} \). The exponential function \( RT = 589 + 294e^{-0.2N} \) was generated by fitting an exponential function to the power function data. The similarity in the learning curves reflects the potential for one to mimic the other.
Logan (1988, 1992) showed that the independent race model predicted a power function speedup on the assumption that the distribution of retrieval times was Weibull. The Weibull is a generalization of the exponential in which the exponent is raised to a power. Its distribution function is

\[ F(x) = 1 - \exp[-wx^c]. \quad (13) \]

Per Gumbel (1958), the distribution function of minima from \( N \) i.i.d. distributions is

\[ F_{\text{min}}(x) = 1 - [1 - F(x)]^N. \quad (14) \]

Substituting Equation (13) into Equation (14) yields

\[ F_{\text{min}}(x) = 1 - (1 - (1 - \exp[-wx^c]))^N = 1 - \exp[-wx^c]^N = 1 - \exp[-Nwx^c] = 1 - \exp[-w(N^{1/c}x)^c] = F(N^{1/c}x). \quad (15) \]

Thus, the distribution of minima of \( N \) samples from i.i.d. Weibull distributions is itself a Weibull distribution with its scale reduced by a power function of \( N \). This implies that the entire distribution of retrieval times decreases as a power function of practice—the mean, the standard deviation, and all of the quantiles of the distribution should all decrease as power functions of practice with a common exponent, \( 1/c \). Logan (1988) tested the prediction for means and standard deviations, and Logan (1992) tested the prediction for distribution. The predictions were mostly confirmed.

Instance theory assumes two races. One, just described, is between the various traces in memory. It determines the speedup in memory retrieval over practice. The other is between the algorithm and memory retrieval. The algorithm is necessary early in practice before instances are available in memory, so subjects are prepared to use it on each trial. If the stimulus is novel, they have no other choice but to execute the algorithm. If the stimulus is familiar, the algorithm and memory retrieval start at the same time, and the faster of the two determines performance. The theory assumes that the time for the algorithm does not change over practice whereas the time for memory retrieval speeds up. This allows memory retrieval to win the race more and more often, until the subject relies on it entirely and abandons the algorithm.

The assumption that the algorithm does not change with practice was made for convenience. With that assumption, the finishing time for the algorithm can be thought of as just another Weibull distribution in the race whose effects will be dominated by other runners as practice continues. Instance theory assumes also that the time to retrieve an individual memory trace does not change over practice. This assumption was made for convenience and for rhetorical force. The “parent” distributions of algorithm finishing time and memory retrieval time do not change with practice. All that changes is the number of traces, and that produces the statistical facilitation that predicts the power law of learning.

Compton and Logan (1991) pitted the independent race model against a probability mixture model in which subjects choose to use the algorithm with probability \( p \) and memory retrieval with probability \( 1 - p \). The parent distributions do not change with practice. Memory retrieval is faster than the algorithm at the outset and remains so throughout practice. Instead, \( p \) changes in a manner that produces the power function speedup required by the power law. This model predicted the same