New Ideas in Psychology 63 (2021) 100873

Contents lists available at ScienceDirect

New Ideas in Psychology

journal homepage: www.elsevier.com/locate/newideapsych

Toward a richer understanding of human cognition: Unleashing the full potential of the concurrent information-processing paradigm

Philip Millroth *

Department of Psychology, Uppsala University, P.O. Box 1225, SE-751 42, Uppsala, Sweden

ARTICLE INFO

Keywords:
Cognitive processes
Information processing
Concurrency

ABSTRACT

One of the most influential working hypotheses in psychology, to this day, is that human information processing in higher-order cognition (e.g., judgment and decision-making) is constrained by having to process objects serially, one at a time. However, a rather large body of research, accumulated over the past 50 years, has demonstrated that serial-processing models provide a poor descriptive account of human information processing. An alternate to the serial-processing view is that people can process information concurrently; many cognitive processes can advance independently of each other even if the system involves only a single central information-processing unit (i.e., central executive). Perhaps this general idea can be advanced beyond its present standing and is to provide new powerful tools in the pursuit of understanding human behavior. To this end, the article provides a review of (*i*) the conceptual differences between different types of processing (serial, concurrent, parallel), (*ii*) recent advancements in the field of computer science, and (*iii*) existing research on human information-processing, which is in line with the advancements in computer science. Finally, the article provides a discussion of outstanding research questions gleaned from these reviews—questions that could stimulate entirely new research programs.

1. Introduction

Psychologists—and more broadly, cognitive scientists—have long relied on using new technologies as a source of abstract metaphors for understanding the human mind (Gigerenzer, 1991). The arguably most influential metaphor is that of the mind as an information processor, much like a computer. The computer metaphor has stimulated some of the most influential work in the cognitive sciences during the past centuries and is still instrumental to researchers (Gigerenzer, 2020; Gigerenzer & Goldstein, 1996). A central tenet of the computer metaphor was—and still is—that higher-order cognition (e.g., judgment and decision-making) is constrained by *serial processing*: Objects are processed one at a time by what is called a central executive or central processing unit (CPU).

For example, although your vision allows you to capture most of the stimuli on this page, many actions must be carried out in sequence before you can understand what is written: Words need to be encoded to some temporary memory storage (i.e., short-term memory, working memory); they need to be combined, and information on how the string of words should be interpreted need to be retrieved from long-term memory. Strictly, under the view of serial processing, each of these encoding and retrieval processes cannot start before another is carried out to completion (see, e.g., Broadbent, 1957). The time it takes for the reading is, in this case, determined by how efficiently the CPU can sequentially squeeze through the processes.

However, during the past 40 years, a large body of evidence (reviewed below) has pointed to the notion that the assumption of serial processing is outdated and that human cognition instead operates concurrently, at least in part: Many processes can advance independently of each other. On this account, the most important factor limiting (or allowing, depending on how human beings are graded) our cognition is not determined in terms of how efficiently the CPU can sequentially squeeze through all the processes but more importantly by how processes are organized, that is, how processes communicate with each other and the CPU to schedule efficient solutions.

The starting point for this article is that the concept of concurrency offers untapped theoretical potential for psychologists and, more broadly, cognitive scientists. However, for this potential to be realized, more scientists need to be aware of (*i*) the conceptual differences between different types of processing (serial, concurrent, parallel), (*ii*)

* Corresponding author:
E-mail address: philip.millroth@psyk.uu.se.

https://doi.org/10.1016/j.newideapsych.2021.100873
Received 23 June 2020; Received in revised form 22 February 2021; Accepted 28 April 2021
Available online 7 May 2021

recent advancements in the field of computer science, and (*iii*) existing research on human information-processing, which aligns with the advancements in computer science.

Just as the serial computer metaphor stimulated an immense amount of new research, future research may benefit from incorporated theoretical and practical advancements in the field of computer science to enhance understanding of the level of concurrency in human cognition. Of course, advancements in computer science do not necessarily need to translate to advances in the cognitive sciences: There is no guarantee per se that concurrent systems developed by computer scientists are a proper analogy for human cognition. However, given the initial success of the serial-processing metaphor and existing research on concurrency in human information processing, it seems a natural starting point from which to derive and test hypotheses.

To aid the realization of this untapped potential, the article is structured as follows: The first section outlines the conceptual differences between serial, concurrent, and parallel processes briefly. The second section discusses advancements in the field of computer science from the last decades, providing readers with an additional conceptual understanding of concurrency in information-processing systems. The third section provides a review of research that has focused on examining concurrency in human cognition. The last section provides a discussion on outstanding research questions in the light of the previous sections to push the field to make new theoretical advances.

2. Defining features of serial, concurrent, and parallel processes

In the large literature on human information processing, the terms concurrent and parallel, are often used interchangeably, pointing to the notion that processes do not have to wait for each other's completion before starting (as in the case of serial processing). However, there is a crucial conceptual difference between concurrency and parallelism (e.g., Breshears, 2009). Consider the case of a computer program that runs only two processes: Process 1 and Process 2 (see Fig. 1). In concurrent processing, the processes advance independently of each other (e.g., the second process does not have to wait until the first is finished for it to advance), but in parallel processing, the processes advance simultaneously. To achieve parallelism, the system needs more than one central information-processing unit (CPU—corresponding to the notion of a central executive), but this is not a requirement for concurrent or serial processing. To have Process 1 and Process 2 in progress at the same time, the program must have started working on both processes but might temporarily switch back and forth between them. To this end, concurrency programs typically divide processes into smaller parts of the overall solution. Of course, a program can be both parallel and concurrent.

To illustrate further the differences between serial, concurrent, and parallel, consider how to go about building the walls of a brick house. If the processing was strictly serial, one would prepare the cement that glues the first row of bricks to the floor, lay out the bricks for the row, wait for the cement to dry, prepare more cement, and so on, until all the rows and walls are finished. In this case, there is no overlapping of the different processes (preparing cement, applying cement, laying out bricks, and letting cement dry). If they were done concurrently, one could prepare all the cement right away, keep it fresh and ready in a large cement mixer; and just as a row of bricks was laid out, one would run to the cement mixer and get enough cement to cover the row, and then go on with laying out the next row of bricks before the cement for the previous row dries. One could even switch back and forth between multiple walls. In this case, there can be some overlapping of the processes, and it is all about organizing the sequence of events in the most efficient order. In contrast, employing a parallel approach would essentially entail hiring personnel where each person is responsible for a given process at a given wall.

3. Concurrency in computer science: past, present, and future

Around the same time that Broadbent (1957) proposed that human cognition relies on serial task scheduling, computer science started to move away from such implementations. As first outlined by Gill (1958), many processes within computer programs could be made to run concurrently within a CPU by using *time-sharing*. Instead of executing one process after the other, two processes could be scheduled side by side, and even if the processes were not executed simultaneously, they could closely mimic parallelism if the moves between the processes were quick enough. In that case, the processes were made concurrently. The idea of time-sharing highlighted the need for *concurrency control*: proper sequencing of the communications between different executions. In other words, if two processes were scheduled side-by-side, how should one decide the optimal way for them to communicate with each other so that when one process has done its part, the other process would be triggered to begin?

3.1. Shared-state models

The initial solution to the problem of concurrency control was proposed in the 1960s by *shared-state models* (see, e.g., Dijkstra, 1959). In these models, information is communicated by having processes (or, rather, *threads*[1]) sharing a state in which memory is shared, and both processes can have access to what the other one writes and reads (see Fig. 2). The problem with this solution is how access to the shared state should be granted; the fact that unnecessary processes should not make use of the limited space, and processes should not occupy the space for too long. Hence, there is a strong need for *synchronization* in order to avoid system errors that can be so severe that the entire program crashes.[2]

However, this is not an issue if the problem that the concurrent program is intended to solve is an easy one (i.e., problems that are so-called "embarrassingly parallel"). An example is Monte Carlo simulations: Here, the general task can be divided into smaller subtasks, and the results of the subtasks are independent of each other. It is only when each subtask is completed that there must be communication between processes. However, many (if not most) problems require a greater deal of communication between processes. In these instances, synchronizing the communication is notoriously hard, and at times, impossible for the shared-state models, even if the programmer is a world-class programmer. Nevertheless, this is the type of concurrency control that is most used today by most programming languages (e.g., Java, C++).

3.2. Message-passing models

An alternative to shared-state memory is *message passing* between processes. Message passing was first proposed by the Actor model (Atkinson & Hewitt, 1977; Hewitt, Bishop, & Steier, 1973), which holds that processes do not share any state with each other but communicate through message passing between each other (see Fig. 3). Here, each process can modify its own state, create more processes, make local

[1] The term *process* is overloaded in the computer science literature on concurrency. In the context of shared-state concurrency, a process refers to an operating system process, which is the execution environment for an entire software application; the concurrent activities are typically referred to as *threads*. In Erlang (Armstrong, 2003), and some other actor -based languages, a process refers to a concurrent activity (i.e., an actor). It is worth noting that an Erlang process is a much smaller unit of computation than an operating system process or a thread; there are examples where a single application is comprised of tens of millions of Erlang processes running concurrently.

[2] It is out of the scope of this study to review the type of error, such as race conditions and deadlocks. For such a review, the technically oriented reader is referred to Lu, Park, Seo, and Zhou (2008).

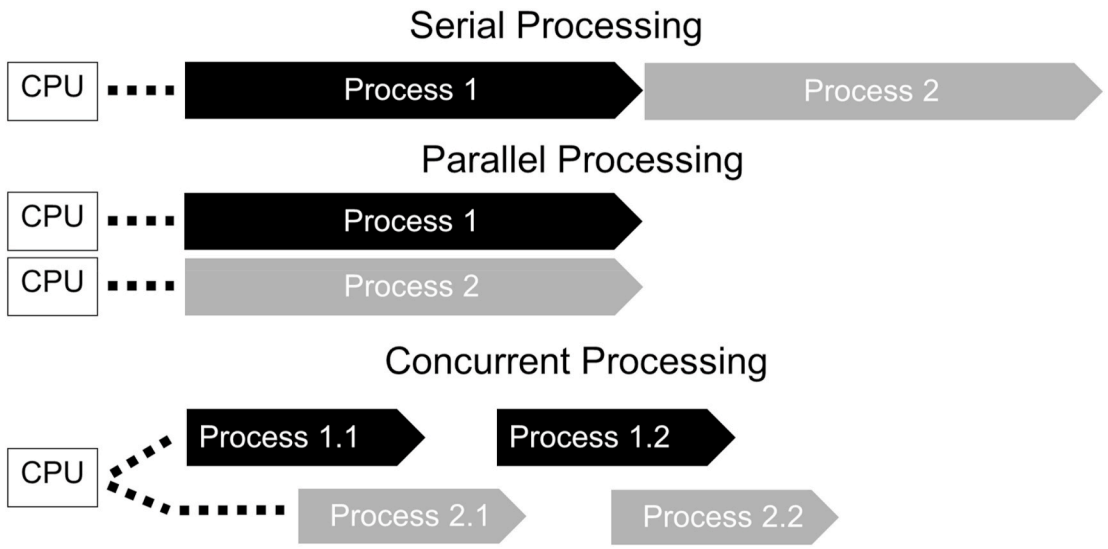

Fig. 1. Illustration of the conceptual differences between *serial*, *parallel*, and *concurrent* processing: In serial processing, each process has to be carried out to completion before the central information processing unit (CPU: sometimes called *the central executive*) can start the next process; in parallel processing, the two processes can be carried out simultaneously because there is more than one single CPU; in the case of concurrent processing, the processes are not carried out exactly simultaneously but are *in progress* at the same time.

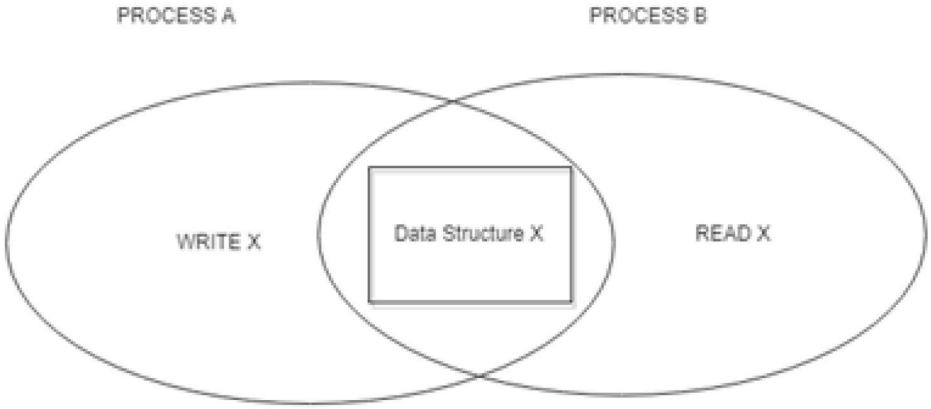

Fig. 2. Illustration of how processes communicate in shared-state models.

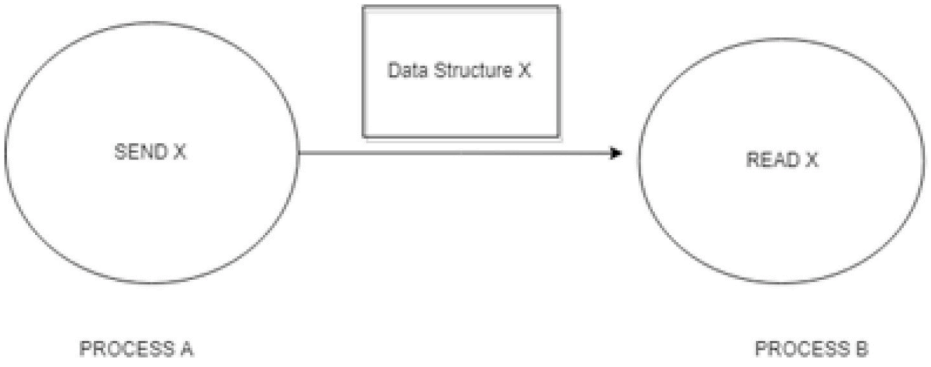

Fig. 3. Illustration of how processes communicate in message-passing models.

decisions, and send and respond to messages. It is important that the *communication be asynchronous*, meaning that communication is unidirectional; each process does not wait for the message to be received by the other. This circumvents the synchronization problems that occur for shared-state models.

Many academic studies have confirmed that message-passing models are superior to shared-state models for problems that are not embarrassingly parallel: They often run faster, are much more code efficient, and are much less error prone (e.g., Armstrong, 2007; Cesarini & Vinoski, 2016; Johansson, Jonsson, Lindgren, Bevemyr, & Millroth, 1997; Nyström, Trinder, & King, 2008). Second, message-passing models have an impressive track record of providing robust and

reliable solutions in the technology industry.[3] Software built on the foundations of message-passing has also proved its worth in academic comparisons with software involving the shared-memory state, excelling at more complex problems while also avoiding errors (Nyström et al., 2008).

4. Human cognition: serial, concurrent, or parallel?

As noted in the Introduction, one of the most influential working hypotheses in cognitive psychology is that higher-order cognition (e.g., judgment and decision-making) is constrained by serial processing. The idea of serial processing was directly translated from how computers at the time (the 1950s) were built and programmed to process information. The idea grew in influence as this technology became more broadly accessible, for example, by becoming available at universities starting in the second half of the 20th century (Gigerenzer, 2020; Gigerenzer & Goldstein, 1996). During that time (the 1950s–1970s), a growing body of research in psychology, with Herbert Simon and Alan Newell at the forefront, argued that researchers could use computers to program simulations mimicking human behavior in judgment and decision-making tasks (e.g., Newell, Shaw, & Simon, 1958; 1979; Newell & Simon, 1972). The idea was that higher-order cognition is constructed by *elementary information processes* (EIPs), such as *Read into working memory*, *Retrieve from working memory,* and *Compare attributes*, and that if one could write a computer program involving such EIPs that mimicked the behavior observed by humans under the approximately same time frame, then this could be taken as face-value evidence for the underlying theory about human cognition. The research paradigm of using computer simulations involving serially processed EIPs was continually developed late into the 20th century (e.g., Payne, Bettman, & Johnson, 1993) and has continued to stimulate cognitive psychology in the 21st century (e.g., Heck & Erdfelder, 2017; Musalem, Montoya, Meißner, & Huber, 2021; Schoemann et al., 2019a; Schoemann et al., 2019b; Štukelj, 2020). The general idea of serial processing is also at the forefront of popular dual-system theories of cognition and holds that analytical thinking is constrained by serial processing (e.g., Kahneman, 2011). In other words, if an "analytic EIP" is being processed at a given time, nothing else can be processed during the period of that process.

4.1. Evidence against the serial-processing hypothesis

It is obvious that human beings have the capacity for some concurrency and/or parallelism; otherwise, our motor functions and perception would constantly interfere with higher-level cognition (e.g., memory encoding and retrieval, knowledge representation, decision making). However, this article is concerned specifically with higher-level cognition, and research in psychology generally points to the notion that higher-level cognition does not involve parallelism since there is only one CPU (i.e., the central executive). Even among proponents who have refuted the idea of strict serial processing, the notion that higher-order cognition relies on a single CPU has been advocated in a wide array of research areas: general information processing (for a review, see Schweickert & Boggs, 1984); multitasking (e.g., Fischer & Plessow, 2015; Paschler, 1994; Salvucci & Taatgen, 2008); memory (e.g., Townsend & Ashby, 1983; Williams, Eidels, & Townsend, 2014); decision making under risk (e.g., Broniatowski & Reyna, 2017; Reyna & Brainerd, 1995), and human performance on non-deterministic polynomial-time problems; np-hard problems (e.g., Best & Simon, 2000).

If a system involves a single CPU but does not process information in a strict serial manner, then it needs to involve concurrency per definition (Fig. 1). The actual scientific evidence that higher-order cognition works involve concurrent rather than serial processing comes in various forms. However, the evidence can roughly be categorized as arguments resting on either (*i*) validated predictions of behavioral experiments, (*ii*) the ability of *cognitive models* to account for empirical data, and (*iii*) models of neural implementations that can be connected to one of the above.

The purpose of the following subsections is not to cover all relevant literature regarding each of these categories but to give the reader an understanding of the difference between each. Thus, the example studies below were chosen with their pedagogical advantages in mind, and the collection of references underlying this article will together serve the purpose of pointing readers in the proper direction in terms of covering the relevant literature. It should also be stated that, in reality, many of the referenced studies can be fitted under several of the subsections; the organization below simply serves a pedagogical purpose. For example, the simple benchmark models described in the first section are, of course, also cognitive models. The distinction between behavioral experiments (4.1.1) and cognitive models (4.1.2) rather refers to the notion that the discussed cognitive models were developed not with the purpose of specifically distinguishing serial from concurrent processing.

4.1.1. Behavioral experiments

Response times and accuracy measures are two means by which experimental psychologists have been able to distinguish serial from concurrent (and/or parallel) processing in behavioral experiments, making minimal model assumptions (for reviews and mathematical proofs of these and similar methods, see Ruthruff, Pashler, & Hazeltine, 2003; Townsend, 1990; Williams et al., 2014).[4]

For an illustration of this, consider the typical memory-scanning task (e.g., Sternberg, 1966) in which the subject first memorizes a short series of symbols, is then shown a test stimulus, and is finally required to decide whether the stimulus is one of the symbols in memory. If additional mental operations are introduced to this experimental design, for example, by introducing irrelevant auditory or visual stimuli at the time of retrieval, they carry the implications that response-time distributions will differ for serial-processing models and for models not assuming strict serial processing.

Depending also on the time constraints under which stimuli are presented, predictions about accuracy can be made. For example, if participants under one condition are given 800 ms to process and respond to five pieces of stimuli that together form the accuracy of the response, that should each take no more than 200 ms to process, they should—under the assumption of serial processing—perform worse than participants under a control condition, who are given 1000 ms to respond. The reason is that the participants given 800 ms processing time would not have processed all the information necessary for a fully accurate response. However, the results from such accuracy manipulations also contradict the serial-processing assumption; participants in the experimental condition do not perform worse than those in the control condition (Townsend, 1990).

4.1.2. Conceptual and process models of cognition

It is important to keep in mind that descriptive models of human cognition often differ in terms of "levels" (e.g., Marr, 1982). For example, some models are developed at a *computational level* of analysis, aiming to describe *what* is computed (e.g., an expected utility in the case of decision-making under risk), whereas other models are developed at

[3] Starting in the 1990s, a new computer language based on message-passing was developed for handling the massively growing amounts of communication in the cell-phone network (Armstrong, 2003). The language, Erlang, has since then been incorporated in numerous applications provide robust and efficient communication solutions in the technology industry and arguably inspired the use of other programming languages based on the Actor model (e.g., Go).

[4] Many of these reviews make a distinction between serial and parallel processing rather than a distinction between serial and concurrent processing. However, reliance on the definitions outlined in Section 2—which I argue are the most widely accepted ones in computer science—will clarify that the studies often focus on concurrency rather than on parallelism.

an *algorithmic level* (outlining the cognitive processes responsible for carrying out the computation in the cognitive and the order in which they operate), or at an *implementational level* (outlining how these processes are implemented in the hardware of the information-processing system; hence, this amounts to a biological level in the case of human beings). In this section, we are primarily concerned with cognitive models stated at the algorithmic level, that is, the question of how cognitive processes carry out the computations stated at the computational level.

The algorithmic level of explanation can be examined at different resolutions, with each resolution using its own set of representations not necessarily mutually exclusive of each other (Bechtel, 1994; Hommel, 2020; McClamrock, 1991; Minsky, 1988, 2007; Singh, 2012). As an analogy, consider that computer software applications often use an application framework that provides structure and templates. The application framework, in turn, uses common libraries provided with the programming language. Depending on the programming language, the application is compiled to machine language or to machine language for a virtual machine. This compiled application then executes in the context of an operating system, which in itself is a layered software system. The software application could never be accomplished without coding different levels of abstraction. Similarly, human information-processing can be examined at different resolutions: from flowcharts outlining the sequence of events to mathematically formalized process models to real-time modeling outlining the sequence of EIPs, with connectionist (neural) networks arguably the most fine-grained resolution.

Cognitive models that involve concurrent processing and that provide seemingly proper accounts of empirical data from behavioral experiments can be found across many different resolutions. For example, as a contrast to the general "dual-systems" account of human cognition where all so-called analytic processes are run serially (e.g., Kahneman, 2011), Broniatowski and Reyna (2017) provided a model developed mathematically and as a flowchart (henceforth *conceptual models*) that involves the concurrent processing of information and that better accounts for peoples' decisions when choosing between risky prospects. Here, decision "biases" (i.e., departures from normative theory) do not come as a result of a failure of analytic thought processes to intervene in intuitive ones but because people encode and retrieve different types of information from numbers (nominal, ordinal, cardinal) processes that are run concurrently.

Of course, such conceptual models ultimately need to be instantiated by some *process model*—a model outlining how the processing unfolds in real time when also accounting for the systems' constraints (Anderson, 2014; Griffiths, Chater, Kemp, Perfors, & Tenenbaum, 2010; Griffiths, Lieder, & Goodman, 2015; Love, 2015; Newell, 1973a, 1973b; Simon, 1978). At the approximate resolution of EIPs where information-processing is simulated in real time instantiated in a control system (e.g., 4CAPS: Just, Carpenter, & Varma, 1999; ACT-R: Anderson, 2009; EPIC: Kieras & Meyer, 1997), concurrent-processing models have, for example, been able to account for how people can engage together in multiple tasks (Kieras, Meyer, Ballas, & Lauber, 2000; Salvucci & Taatgen, 2008:; Fechner, Schooler, & Pachur, 2017).

Concurrent-processing accounts—though, arguably most prominently at the resolution of connectionist (neural) networks—are a level of resolution aching to that of neurons in the brain. Here, processing involves the propagation of activation among simple units linked to each other through weighted connections where each unit then transmits its activation level to other units in the network by means of its connections to those units (for more detailed conceptual discussions on the mechanics of networks and how they have evolved, see, e.g., Baronchelli, Ferrer-i-Cancho, Pastor-Satorras, Chater, & Christiansen, 2013; Kriegeskorte & Douglas, 2018; Thomas & McClelland, 2008). Important to keep in mind for the present purposes is that the spread of activation among nodes is assumed to be fully sufficient for the processing of information, thus bypassing the need for a central executive to organize and execute the processing of objects (Böttinger, 2015). The amount of concurrent processing involved in any given situation is thus a function of how networks are organized and whether clusters of units are given conflicting information (Shenhav et al., 2017). The parallel constraint satisfaction (PCS) model, e.g., Glöckner and Betsch (2008), offered a particularly telling example of how network approaches have provided concurrent-processing accounts of human behavior, again by comparison to the general "dual-systems" account of human cognition (Kahneman, 2011). In PCS, cognitive processes underlying intuitive and analytic thinking can overlap as "intuitive" units connect with "analytic" units. Thus, on this account, intuitive and analytic thinking can – in part – be occurring concurrently.

4.1.3. Neural implementations

Arguably, most neurocognitive evidence supporting the idea of concurrent information processing comes from research on how neurons in the brain are organized and how they communicate (for a review, see Shenhav et al., 2017). This is to be expected, given that network explanations expressed at the algorithmic level most often build theories using a bottom-up approach, starting with knowledge about how the physical correlates in the brain seem to operate. Albeit simplified, it could in principle be argued that evidence favoring network explanations at the algorithmic level at the same time also amounts to neurocognitive evidence, at least indirectly. Conversely, however, it can be argued that the type of network used today not only to explain human behavior but also to create artificial intelligence is so immensely complex that they amount to the type of black boxes that cognitive psychologists wanted to avoid in the first place.[5] In any case, the descriptive validity of neural networks—algorithmically expressed or as physical entities in the human brain—is the subject of lengthy discussions that are outside the scope of this article (for these purposes, see, e.g., Sternberg, Sternberg, & Mio, 2012). Similarly, researchers have claimed that so-called control systems (i.e., *production systems* or *cognitive architectures*) expressed at the approximate resolution of EIPs were developed with the physical reality of the human brain in mind (Anderson, 2009; Just et al., 1999; Kieras & Meyer, 1997); hence they have also held that these explanations have embedded concurrent information processing as an assumption because "that is how the brain works.''

Admittedly, the degrees of freedom in interpreting neurocognitive data are daunting, making it hard to translate findings at the neurological level with algorithmic-level explanations expressed at a different resolution or even at a corresponding resolution (e.g., Flournoy et al., 2020). However, technological advancements within neuroscience have progressed at record speed in the last decades, and increasingly research seems to focus specifically on combining experimental manipulations

[5] The descriptive validity of neural networks—algorithmically expressed or as physical entities in the human brain—is the subject of lengthy discussions that are out of scope of this article (for a very recent discussion, see Taylor & Taylor, 2020).

and measurement of neurocognitive data to compare the serial vs. concurrent processing models, seemingly favoring the idea that concurrent information processing underlies higher-order cognition. However, that scheduling of processes can be disturbed so that behavior mimics that of a serial processing model (for discussions, see, e.g., Maclean, McSkimming, & McMillan, 2020; Sigman & Dehaene, 2008; Snell & Grainger, 2019; White, Boynton, & Yeatman, 2019; Zylberberg, Slezak, Roelfsema, Dehaene, & Sigman, 2010).

5. From here on: dispersing the mental shackles of the serial-processing assumption

To be clear, the point here is not to bash research grounded in the assumption of serial-processing or to argue that research focused on concurrent processing has been narrow-minded. Rather, the idea underpinning this final section is that the field of psychology may now be in a position to pose completely different questions thanks to previous research. Instead of asking whether the mind processes information in a serial, parallel, or concurrent manner, we may now zoom in on questions that arise under the assumption that higher-order cognition *is* reliant on a concurrent information-processing system. For example: Where is there potential for concurrency in human information processing? How does the information-processing system invoke concurrency control? Of course, these two questions alone do not exhaust the potential of the concurrency concept; they are, however, pedagogical in illustrating the type of discussion that need to take an incremental role in the academic debate about human information processing.

The discussion below will primarily focus on how algorithmic information-processing accounts can be flexibly expressed at the approximate resolution of EIPs. This should not be interpreted as a justified disregard for explanations expressed at the resolution of connectionist (neural) networks. In part, it reflects the fact that the present author's main theoretical and programming competence is not within the realm of neural networks. It is thus the hope that talented network scientists will provide complementary discussions to this one. However, more importantly, it reflects the present author's beliefs that accounts of human behavior provided at the approximate resolution of EIPs are likely to be instrumental for understanding human cognition just as they have been historically (see, e.g., Newell & Simon, 1972; Anderson, 1996). However, currently available tools (e.g., cognitive architectures, such as 4CAPS, ACT-R, EPIC) typically come with a long list of hard-coded theoretical assumptions—specifically about the degree and nature of concurrency—which may thus not be suitable for the purpose of *exploring* concurrency.

If the original serial EIP framework (e.g., Newell et al., 1958; Payne et al., 1993; Simon, 1979) is reprogrammed to allow constraint-free handling of EIPs (e.g., as exemplified in simulations below), this "resolution" at the algorithmic level of analysis has numerous unique advantages (again, according to the present author's opinion). First, such models can easily test constraints stated by conceptual models while keeping the degrees of freedom at a tolerable level. Second, they have the potential to serve as a "middle man" between conceptual models and more fine-grained models (e.g., network models). Third, they can offer rather precise predictions about where in the process a specific experimental manipulation should be targeted. For example, if memory retrieval is the issue of interest, then the manipulation may be targeted when the EIP "retrieval" is predicted to occur. Finally, the switch between EIPs and the effects of manipulations on the process schedule can be connected to biological implementations in real time by using skin-conductance, electroencephalography (EEG), and functional magnetic resonance imaging (fMRI).

5.1. Where is potential for concurrency?

To examine the degree to which concurrency plays a role in specific mental operations, one must first ask where concurrency can be found, or rather, where it cannot be found. In a sense, this is like the process of constructing computer software: A computer programmer developing a software does not start by asking "How can I make the processes underpinning this program run concurrently" but by asking "*Which* processes *can* be made to run concurrently?"

If one wishes to model human behavior, theoretical considerations from cognitive psychology will, of course, be important: Which processes must finish before other processes can start and why (for a similar discussion, see Shin & Rosenbaum, 2002)? However, any research field needs, from time to time, to complement the confirmatory approach (prediction driven) with an exploratory approach (Behrens, 1997; Judd & Kenny, 2010; Kelder, Conklin, Evelo, & Pico, 2010) in part because exploratory approaches help us avoid sampling bias (see, e.g., Fiedler, 2011; Koch, Imhoff, Dotsch, Unkelbach, & Alves, 2016). In other words, to evaluate the support for a given hypothesis, we need to evaluate it compared to competing hypotheses that are simultaneously tested. Given that so much of our existing hypotheses are grounded either in terms of serial processing, or in specific implementations of concurrency solutions, it seems timely to allow a more flexible modeling approach.

To illustrate, let us consider how the execution of a decision-making strategy could be structured with varying degrees of concurrency. The focus will be on adapting a decision-making strategy, LEX, identified to be commonly used by people in risky decision-making (Payne, Bettman, & Johnson, 1988), and show how elements of this strategy can be made concurrent.

LEX works as follows: Consider the choices set out below:

Alternative	Probability of outcome	
	.75	.25
A	$100	$400
B	$200	$0

First, the most important attribute for each alternative is determined according to the highest probability of occurrence (i.e., "0.75 to gain $100" for A, and "0.75 to gain $200" for B). Thereafter, the payoffs of these two attributes are compared (i.e., "$100" for A, vs. "$200" for B), and the alternative with the highest payoff is chosen (B in this case). If there are ties, the second most important attribute is examined, and so on until the tie is broken. To execute this algorithm, three different EIPs are used: MOVE (directs attention to the next highest probability or payoff), READ (reads the next highest probability or payoff and stores it in working memory), and COMPARE (compares the payoffs of the identified attributes).

Throughout this section, the same implementation of LEX will be used. It is an implementation in which LEX is executed to make decisions in an environment where an artificial agent has to choose between four different alternatives, each with four possible outcomes. These levels were chosen to make the effects of concurrency evident. For the illustrative purposes, the models are kept free of any additional modeling of motor movements; hence, the focus is merely on the cognitive processes involved. The length of each EIP was chosen in accordance with previous literature on similar subjects (e.g., Fechner et al., 2017; Johnson & Payne, 1985; Payne et al., 1988)[6-7].

Because the present simulation is merely illustrative, the following three dependencies have been set only to provide an example. First, a READ cannot start until its related MOVE is finished (the information

[6] MOVE: 50 ms; READ: 200 ms; COMPARE: 300 ms; RETRIEVE (used in simulations further below): 20 ms; ASSOCIATE (used in simulations further below): 200 ms.

[7] The time between EIPs (i.e., when the program performed input/output and various other kinds of internal book-keeping) constituted around 1 ms at the most, thus a negligible part of the implementations. All program codes are available online, along with instructions on how to run the programs: https://osf.io/7m3te/.

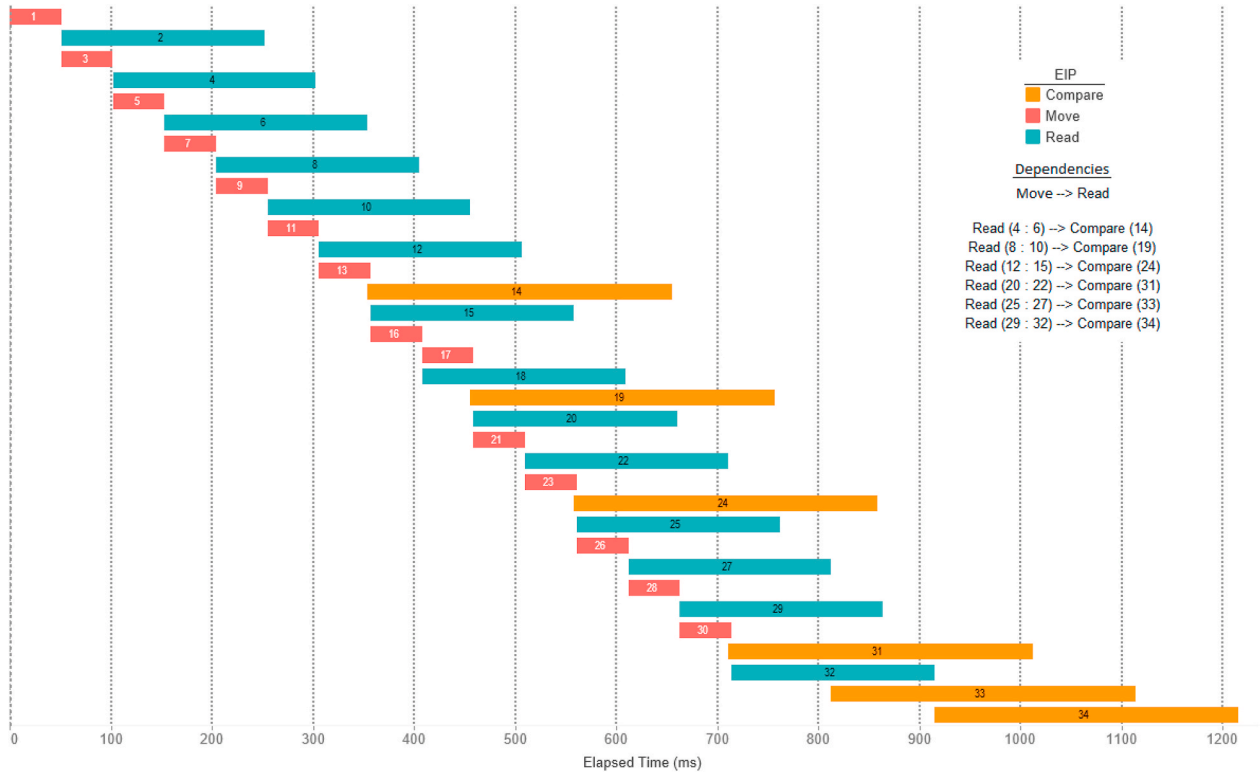

Fig. 4. Process schedule for the concurrent version of LEX. Numbers inside the bars show starting order of the processes. Dependencies in the legend signal which process must wait for which process to finish before starting (e.g., a Move has to be completed before a Read can start; the Compare with starting order 14 has to wait for two Reads with starting orders 4 and 6).

has to be attended to before it can be read into memory). Second, two READs have to be completed before a COMPARE can start (the necessary items have to be in order for a comparison to be made). Third, two MOVEs cannot run parallelly (it is assumed that attention cannot be directed at two places simultaneously). Apart from that, the artificial agent has endless capacities to run processes concurrently.

To appreciate the nature of any concurrent program fully, a process schedule that illustrates the program from start to end is necessary. The process schedule for the concurrent version is illustrated in Fig. 4, where each process is depicted in real time along with the set dependencies. It took the artificial agent 1216 ms to complete the strategy—4.39 times faster than the serial version.[8]

To depict further the important role of concurrency, we consider a more complex version of LEX. Two additional EIPs were added to illustrate a very simple noise process: RETRIEVE and ASSOCIATE. These EIPs capture that the artificial agent has had the bad fortune that irrelevant information is retrieved from long-term memory each time that new information is attended to (RETRIEVE), and that this is read into working memory where it is being associated with the other information that is coming in from the other process (ASSOCIATE). The process schedule for the concurrent version is depicted in Fig. 5. For the serial version, it took the artificial agent 8442 ms to complete the strategy (1.58 times slower than the serial version without the noise process). However, for the concurrent version, it took the agent 1215 ms to complete the strategy, meaning that it is virtually unaffected by the added processes while being 6.95 times faster than the serial version with the noise process. Thus, how much effect the noise will have depend on the theoretical assumption one makes about the limitations of processing channels, and one could easily think of other process schedules to be compared with that in Fig. 5.

The overarching point is that various models can easily be construed and compared with empirical data. As a mental exercise, consider a hypothetical scenario where we collect behavioral data in which experiment participants are taught to apply the LEX rule, and the results show that most people complete the strategy in approximately 1220 ms even when we simultaneously introduce stimuli intended to make participants generate irrelevant associations. At the same time, competing models, instantiated in a theoretically constrained hypothesis spaces, predict that it should take approximately 4000 ms without the introduction and 6000 ms with noise. This would not entail that the more unconstrained model is *the* proper explanation of the observed behavior, but it does entail that a current theoretical model may need to revise specific assumptions. Intriguingly, it is a recurrent observation that proposed process models overestimate the time it takes people to complete decision strategies (e.g., Fechner et al., 2017; Payne et al., 1988).

5.2. How does the human information-processing system invoke concurrency control?

Section 3 introduced the issue of *concurrency control*: proper sequencing of the communication between different executions (i.e., the optimal way for them to communicate with each other so that when one process has performed its part, the other process is triggered when to

[8] Here, speedup is simply calculated as the ratio between the serial version and the concurrent version. However, if different concurrent versions are compared to generate a mapping of how human concurrent systems are constructed, it may also be of interest to measure *efficiency:* how well software utilizes the computational resources of the system. To calculate the efficiency of parallel execution, the observed speedup is divided by the number of cores (i.e., processors), and this number is then expressed as a percentage. For example, a 53X speedup on 64 cores equates to an efficiency of 82.8% (53/64 = 0.828). This means that, on average, over the course of the execution, each core is idle for about 17% of the time. Also, the realized speedup can be compared with the theoretically estimated speedup determined by Amdahl's law. For further details on the topic, see Woo and Lee (2008).

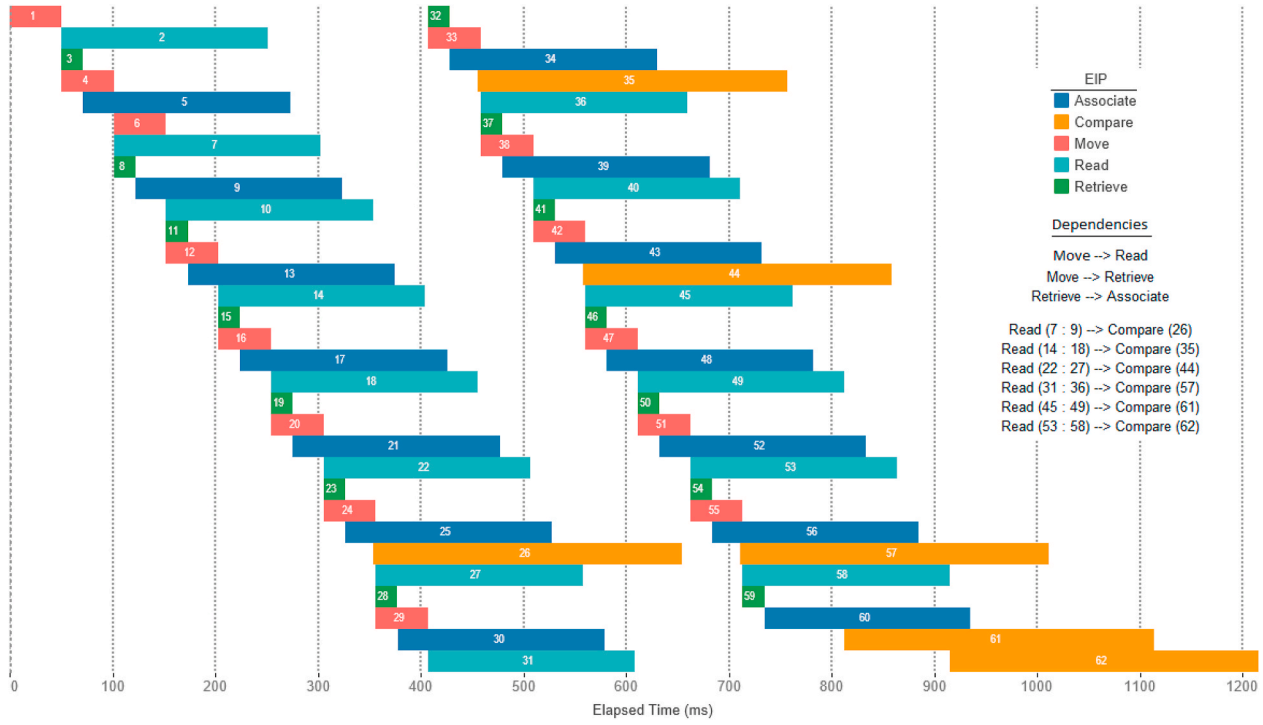

Fig. 5. Process schedule for the concurrent version of LEX that implements two additional EIPs (Retrieve; Associate). Numbers inside the bars show starting order of the processes. Dependencies in the legend signal which process must wait for which process to finish before starting (e.g., a Move has to be completed before a Read can start; the Compare with starting order 26 has to wait for two Reads with starting orders 7 and 9).

begin). Section 3 further outlined that computer scientists have provided two ways of achieving concurrency control: (*i*) by *shared-state* solutions where information is communicated by having processes sharing a state in which memory is shared, and both processes can have access to what the other one writes and reads, or (*ii*) by *message-passing* solutions where processes do not share any state but communicate through message-passing between each other.

Although Hewitt (1977), and indirectly Minsky (1988; 2007), stressed that researchers interested in modeling human-like intelligence would need to move away from shared-state models and toward message-passing models, cognitive researchers have largely avoided the issue, which has led them seemingly to settle for shared-state solutions (e.g., Salvucci & Taatgen, 2008). However, this avoidance of message-passing solutions is likely not a result of active choice but a reflection of the notion that behavioral researchers often transfer the currently available technical tools into their theories (Gigerenzer, 2020): Shared -state solutions have been predominant in computer-software applications, often directly underpinning programming languages (e.g., C++, Java).

Message-passing models have recently proved successful in bottom-up network approaches (i.e., more detailed resolution at the algorithmic level of explanation) to create intelligent agents through the process of neuro-evolution (e.g., Sher, 2013), and there is nothing hindering top-down approaches. For example, the simulations in this section were conducted by using the programming language Erlang (Armstrong, 2003), in which communication between processes is asynchronous (i.e., communication only flows in one direction). Handling both top-down and bottom-up approaches is indeed an important feature for any framework aiming to model human cognition (Taatgen, 2005). Of course, as stated in the Introduction, there is no guarantee per se that concurrency solutions developed by computer scientists are proper analogies for human cognition, and the reader may rightfully wonder whether the difference between shared-state and message-passing solutions is foremost a programming issue rather than a possible central focus of empirical inquiry. After all, the simulations provided above could also be programmed by using shared-state solutions, with no noticeable difference in program runtime.

However, it *is* of the essence that cognitive scientists determine which solution the human information-processing system uses to invoke concurrency control. The two solutions carry contrasting implications: From shared-state solutions follows a depicting image of an information-processing system easily subject to interference while message-passing solutions push the boundaries, suggesting that human beings is able to process much more information in a robust manner than we currently depict. Naturally, these contrasting views on how human beings process information have implications for how we design the environments in which we engage. Of course, as evident from the foregoing literature reviews, it is clear that there are constraints to how much information people can process concurrently. However, settling the issue of shared-state vs. message-passing has the potential to allow more tailored information structuring in the environments in which people engage (e.g., digital platforms in schools and workplaces, marketing).

A potential starting point for empirical inquiries on the issue involves testing the descriptive validity of shared-state models and message-passing models, similar to the simulations provided above. However, simply comparing response times will not suffice. The focus should be on how the two types of models differ in their *ability* to handle increasing degrees of concurrency *efficiently and robustly*. As more and more processes have to be synchronized, computer simulations of the two will increasingly differ in program length, programming effort, runtime efficiency, memory consumption, and reliability (for a discussion on how to perform empirical comparisons of computer programs by using different programming solutions, see Prechelt, 2000). Do these differences measure up to similar differences in the output observed for human beings (e.g., quantity of decision biases, energy resources in the brain, eye movements)? Of course, the role of the programmer is crucial, arguably demanding that the topic be addressed using cross-laboratory collaborations.

6. Conclusions

Unleashing the full potential of the concurrency concepts offers the promise of advancing new perspectives on human information processing mechanisms. New accounts that focus specifically on concurrency may prove to yield novel insights regarding the bounds and capabilities of the human mind. The longer-term goals should, however, not be to provide merely efficient algorithms for cognitive modeling but to provide tools that allow full modeling of empirical data coupled with the potential for simulation. It will be important that researchers simply do not fit their algorithms to data and make post hoc explanations that explain the match between algorithm and observed behavior. Instead, researchers should compare multiple models, varying in their degree of theoretical constraints: Some models may be strongly theoretically constrained by having them instantiated in existing cognitive architectures while alternative models may be derived from advances in computer science and with less involvement of presupposed cognitive constraints.

Author statement

Philip Millroth: All work related to the article was conducted by this sole author.

Acknowledgment

Thanks to Håkan Millroth for helping with the Erlang-implementation. The research was supported by the Swedish Research Council and The Ryoichi Sasakawa Young Leaders Fellowship Fund.

References

Anderson, J. R. (1996). ACT: A simple theory of complex cognition. *American Psychologist, 51*(4), 355.
Anderson, J. R. (2009). *How can the human mind occur in the physical universe?* Oxford University Press.
Anderson, J. R. (2014). *Rules of the mind*. Psychology Press.
Armstrong, J. (2003). *Making reliable distributed systems in the presence of software errors (Doctoral dissertation)*.
Armstrong, J. (2007). A history of Erlang. In *Proceedings of the third ACM SIGPLAN conference on history of programming languages*. ACM, 6-1.
Atkinson, R., & Hewitt, C. (1977). January). Synchronization in actor systems. In *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages* (pp. 267–280).
Baronchelli, A., Ferrer-i-Cancho, R., Pastor-Satorras, R., Chater, N., & Christiansen, M. H. (2013). Networks in cognitive science. *Trends in Cognitive Sciences, 17*(7), 348–360. Bechtel, 1994.
Behrens, J. T. (1997). Principles and procedures of exploratory data analysis. *Psychological Methods, 2*(2), 131.
Best, B. J., & Simon, H. A. (2000). Simulating human performance on the traveling salesman problem. In *Proceedings of the third international conference on cognitive modeling* (pp. 42–49).
Breshears, C. (2009). *The art of concurrency: A thread monkey's guide to writing parallel applications*. O'Reilly Media, Inc.
Broadbent, D. E. (1957). A mechanical model for human attention and immediate memory. *Psychological Review, 64*(3), 205.
Broniatowski, D. A., & Reyna, V. F. (2018). A formal model of fuzzy-trace theory: Variations on framing effects and the Allais paradox. *Decision, 5*(4), 205.
Cesarini, F., & Vinoski, S. (2016). *Designing for Scalability with Erlang/OTP: Implement robust, Fault-Tolerant systems*. O'Reilly Media, Inc.
Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*(1), 269–271.
Fechner, H. B., Schooler, L. J., & Pachur, T. (2018). Cognitive costs of decision-making strategies: A resource demand decomposition analysis with a cognitive architecture. *Cognition, 170*, 102–122.
Fiedler, K. (2011). Voodoo correlations are everywhere—not only in neuroscience. *Perspectives on Psychological Science, 6*(2), 163–171.
Fischer, R., & Plessow, F. (2015). Efficient multitasking: Parallel versus serial processing of multiple tasks. *Frontiers in Psychology, 6*, 1366.
Gigerenzer, G. (1991). From tools to theories: A heuristic of discovery in cognitive psychology. *Psychological Review, 98*(2), 254.
Gigerenzer, G. (2020). How to explain behavior? *Topics in Cognitive Science, 12*(4), 1363–1381.
Gigerenzer, G., & Goldstein, D. G. (1996). Mind as computer: Birth of a metaphor. *Creativity Research Journal, 9*(2–3), 131–144.
Gill, S. (1958). Parallel programming. *The Computer Journal, 1*(1), 2–10.
Glöckner, A., & Betsch, T. (2008). Modeling option and strategy choices with connectionist networks: Towards an integrative model of automatic and deliberate decision making. *MPI Collective Goods*. Preprint, (2008/2).
Griffiths, T. L., Chater, N., Kemp, C., Perfors, A., & Tenenbaum, J. B. (2010). Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in Cognitive Sciences, 14*(8), 357–364.
Griffiths, T. L., Lieder, F., & Goodman, N. D. (2015). Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in Cognitive Science, 7*(2), 217–229.
Heck, D. W., & Erdfelder, E. (2017). Linking process and measurement models of recognition-based decisions. *Psychological Review, 124*(4), 442.
Hewitt, C., Bishop, P., & Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd international joint conference on Artificial intelligence* (pp. 235–245).
Hommel, B. (2020). Pseudo-mechanistic explanations in psychology and cognitive neuroscience. *Topics in Cognitive Science, 12*(4), 1294–1305.
Johansson, E., Jonsson, C., Lindgren, T., Bevemyr, J., & Millroth, H. (1997). A pragmatic approach to compilation of Erlang. In *International symposium on programming language implementation and Logic programming* (pp. 419–420). Berlin, Heidelberg: Springer.
Johnson, E. J., & Payne, J. W. (1985). Effort and accuracy in choice. *Management Science, 31*(4), 395–414.
Judd, C. M., & Kenny, D. A. (2010). Data analysis in social psychology: Recent and recurring issues. *Handbook of Social Psychology, 1*, 115–139.
Just, M. A., Carpenter, P. A., & Varma, S. (1999). Computational modeling of high-level cognition and brain function. *Human Brain Mapping, 8*(2-3), 128–136.
Kahneman, D. (2011). *Thinking, fast and slow*. Macmillan.
Kelder, T., Conklin, B. R., Evelo, C. T., & Pico, A. R. (2010). Finding the right questions: Exploratory pathway analysis to enhance biological discovery in large datasets. *PLoS Biology, 8*(8), Article e1000472.
Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12*(4), 391–438.
Kieras, D. E., Meyer, D. E., Ballas, J. A., & Lauber, E. J. (2000). Modern computational perspectives on executive mental processes and cognitive control: Where to from here. *Control of cognitive processes: Attention and Performance, XVIII*, 681–712.
Koch, A., Imhoff, R., Dotsch, R., Unkelbach, C., & Alves, H. (2016). The ABC of stereotypes about groups: Agency/socioeconomic success, conservative–progressive beliefs, and communion. *Journal of Personality and Social Psychology, 110*(5), 675.
Kriegeskorte, N., & Douglas, P. K. (2018). Cognitive computational neuroscience. *Nature Neuroscience, 21*(9), 1148–1160.
Love, B. C. (2015). The algorithmic level is the bridge between computation and brain. *Topics in Cognitive Science, 7*(2), 230–242.
Lu, S., Park, S., Seo, E., & Zhou, Y. (2008). Learning from mistakes: A comprehensive study on real world concurrency bug characteristics. *ACM SIGOPS - Operating Systems Review, 42*(2), 329–339.
Maclean, L. M., McSkimming, P., & McMillan, T. M. (2020). The association between dual-task walking and counting responses and cognitive function and disability after severe head injury: A preliminary study. *Neuropsychological Rehabilitation*, 1–13.
Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. Cambridge, Massachusetts: MIT Press.
McClamrock, R. (1991). Marr's three levels: A re-evaluation. *Minds and Machines, 1*(2), 185–196.
Minsky, M. (1988). *Society of mind*. Simon and Schuster.
Minsky, M. (2007). *The emotion machine: Commonsense thinking, artificial intelligence, and the future of the human mind*. Simon and Schuster.
Musalem, A., Montoya, R., Meißner, M., & Huber, J. (2021). Components of attentional effort for repeated tasks. *Journal of Behavioral Decision Making, 34*(1), 99–115.
Newell, A. (1973a). *You can't play 20 questions with nature and win: Projective comments on the papers of this symposium*.
Newell, A. (1973b). Production systems: Models of control structures. In *Visual information processing* (pp. 463–526).
Newell, A., Shaw, J. C., & Simon, H. A. (1958). Elements of a theory of human problem solving. *Psychological Review, 65*(3), 151.
Newell, A., & Simon, H. A. (1972). *Human problem solving* (Vol. 104). Englewood Cliffs, NJ: Prentice-hall, 9.
Nyström, J. H., Trinder, P. W., & King, D. J. (2008). High-level distribution for the rapid production of robust telecoms software: Comparing C++ and ERLANG. *Concurrency and Computation: Practice and Experience, 20*(8), 941–968.
Pashler, H. (1994). Dual-task interference in simple tasks: Data and theory. *Psychological Bulletin, 116*(2), 220.
Payne, J. W., Bettman, J. R., & Johnson, E. J. (1988). Adaptive strategy selection in decision making. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 14*(3), 534.
Payne, J. W., Bettman, J. R., & Johnson, E. J. (1993). *The adaptive decision maker*. Cambridge University Press.
Prechelt, L. (2000). An empirical comparison of seven programming languages. *Computer, 33*(10), 23–29.
Reyna, V. F., & Brainerd, C. J. (1995). Fuzzy-trace theory: An interim synthesis. *Learning and Individual Differences, 7*(1), 1–75.
Ruthruff, E., Pashler, H. E., & Hazeltine, E. (2003). Dual-task interference with equal task emphasis: Graded capacity sharing or central postponement? *Perception & Psychophysics, 65*(5), 801–816.
Salvucci, D. D., & Taatgen, N. A. (2008). Threaded cognition: An integrated theory of concurrent multitasking. *Psychological Review, 115*(1), 101.

Schoemann, M., Schulte-Mecklenbeck, M., Renkewitz, F., & Scherbaum, S. (2019). Forward inference in risky choice: Mapping gaze and decision processes. *Journal of Behavioral Decision Making, 32*(5), 521–535.
Schweickert, R., & Boggs, G. J. (1984). Models of central capacity and concurrency. *Journal of Mathematical Psychology, 28*(3), 223–281.
Shenhav, A., Musslick, S., Lieder, F., Kool, W., Griffiths, T. L., Cohen, J. D., et al. (2017). Toward a rational and mechanistic account of mental effort. *Annual Review of Neuroscience, 40*, 99–124.
Sher, G. I. (2013). Developing a feed forward neural network. In *Handbook of Neuroevolution through Erlang* (pp. 153–185). New York, NY: Springer.
Shin, J. C., & Rosenbaum, D. A. (2002). Reaching while calculating: Scheduling of cognitive and perceptual-motor processes. *Journal of Experimental Psychology: General, 131*(2), 206.
Sigman, M., & Dehaene, S. (2008). Brain mechanisms of serial and parallel processing during dual-task performance. *Journal of Neuroscience, 28*(30), 7585–7598.
Simon, H. A. (1978). Information-processing theory of human problem solving. *Handbook of Learning and Cognitive Processes, 5*, 271–295.
Simon, H. A. (1979). Information processing models of cognition. *Annual Review of Psychology, 30*(1), 363–396.
Singh, P. (2012). Examining the society of mind. *Computing and Informatics, 22*(6), 521–543.
Snell, J., & Grainger, J. (2019). Consciousness is not key in the serial-versus-parallel debate. *Trends in Cognitive Sciences, 23*(10), 814–815.
Sternberg, S. (1966). High-speed scanning in human memory. *Science, 153*(3736), 652–654.
Sternberg, R. J., Sternberg, K., & Mio, J. (2012). *Cognitive psychology*. Cengage Learning Press.
Štukelj, G. (2020). On the simplicity of simple heuristics. *Adaptive Behavior, 28*(4), 261–271.
Taatgen, N. (2005). Modeling parallelization and flexibility improvements in skill acquisition: From dual tasks to complex dynamic skills. *Cognitive Science, 29*(3), 421–455.
Taylor, J. E. T., & Taylor, G. W. (2020). Artificial cognition: How experimental psychology can help generate explainable artificial intelligence. *Psychonomic Bulletin & Review*, 1–22.
Thomas, M. S., & McClelland, J. L. (2008). *Connectionist models of cognition*.
Townsend, J. T. (1990). Serial vs. parallel processing: Sometimes they look like Tweedledum and Tweedledee but they can (and should) be distinguished. *Psychological Science, 1*(1), 46–54.
Townsend, J. T., & Ashby, F. G. (1983). *Stochastic modeling of elementary psychological processes*. CUP Archive.
White, A. L., Boynton, G. M., & Yeatman, J. D. (2019). You can't recognize two words simultaneously. *Trends in Cognitive Sciences, 23*(10), 812–814.
Williams, P., Eidels, A., & Townsend, J. T. (2014). The resurrection of Tweedledum and Tweedledee: Bimodality cannot distinguish serial and parallel processes. *Psychonomic Bulletin & Review, 21*(5), 1165–1173.
Woo, D. H., & Lee, H. H. S. (2008). Extending Amdahl's law for energy-efficient computing in the many-core era. *Computer, 41*(12), 24–31.
Zylberberg, A., Slezak, D. F., Roelfsema, P. R., Dehaene, S., & Sigman, M. (2010). The brain's router: A cortical network model of serial processing in the primate brain. *PLoS Computational Biology, 6*(4), Article e1000765.