4-2019

Cognitive Sciences for Computing Education

Anthony V. Robins
University of Otago

Lauren Margulieux
Georgia State University

Briana B. Morrison
University of Nebraska at Omaha

Cognitive sciences for computing education

Anthony V. Robins, Lauren E. Margulieux, Briana B. Morrison

1 Introduction

Cognitive Science is typically defined as the interdisciplinary study of the mind and its processes. The field emerged in the 1950s, driven largely by progress in cognitive psychology and artificial intelligence, see Gardner (1985) for a history. The disciplines usually included within its scope include philosophy, neuroscience, psychology, anthropology, linguistics, and artificial intelligence.

The field of Computing Education Research (CEdR)[1] is an example of Discipline-Based Education Research (Singer, Nielsen & Schweingruber, 2012). It has developed within the context of, and been shaped by, cognitive science and education. **Methodologically**, CEdR shares the tools and methods of enquiry employed within these disciplines (see Chapters 2.1, 2.2, 2.3, 2.4). **Empirically**, CEdR has been guided and influenced by what we know about human cognition and learning. During the 1970s to 80s the emerging CEdR field was characterised by a focus on "the psychology of programming", and it adopted findings such as the limitations of working memory, and the importance of the knowledge structures of schemata / plans (see Chapters 1.2 and 3.1). **Theoretically**, CEdR adopted paradigms such as cognitivism and (later) constructivism (see Chapters 2.5, 2.7, 3.4). In recent decades it has continued adopting and exploring theoretical frameworks like developmental stages, mental models and cognitive load (Chapter 3.1).

Arguably, however, research within CEdR would benefit from further and ongoing exploration of foundational cognitive science. In 2005 the Special Interest Group in Computer Science Education (SIGCSE) held a panel discussion on Challenges to Computer Science Education Research. Panellists commented on the "isolation" of CEdR, and that "Too much of the research in computing education ignores the hundreds of years of education, cognitive science, and learning sciences research that have gone before us" (Almstrum *et al.*, 2005). Despite progress since 2005, this criticism remains relevant today. This chapter is intended to be a

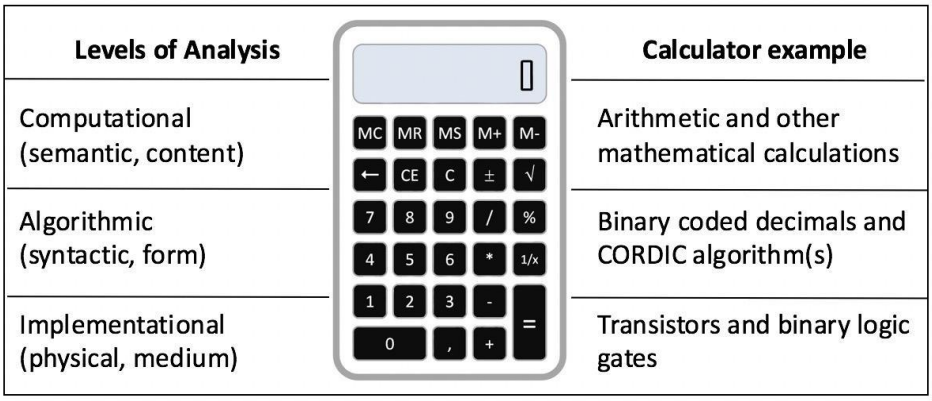
[1] Frequently used acronyms are listed at the end of the chapter.

contribution towards addressing it, by summarising some of the findings of these "hundreds of years" of research as they pertain to human cognition and learning, and by drawing out the implications for CEdR. This is not an easy task, and readers familiar with particular topics will see that we have summarised heavily. We aim to provide an introduction to foundational concepts with a roadmap to the broader literature (sections 2 to 5), and to highlight two areas of particular interest (sections 6 and 7).

2 Brains, computers and levels of analysis

In the most general sense, both the brain and the computer process information. In the brain, various patterns of neural activation give rise to emergent phenomena that we experience and describe as sensations, thoughts, memories and the like. Our memories are made up of overlapping patterns that are approximately recreated in the process of recall, and our reasoning is determined by heuristics, expectations, and mental models of real or imagined worlds. In the computer a central processor reads and writes binary patterns that we interpret as symbols from and to specified locations in a passive memory. Various sequences of symbolic manipulation give rise to emergent phenomena that we describe as computations, processes, and other forms of output behaviour.

Cognitive science generally shares the methods and frameworks of its component disciplines. There is, however, one **theoretical** insight which arose within the field that we should draw attention to, that is Marr's concept of "levels of analysis" (Marr & Poggio, 1976; Marr, 1982). Marr proposes that explanations of the functioning of information processing systems should consider three distinct but complementary levels: computational, algorithmic, and implementational. The *computational level* describes what the system does, its purpose and the tasks that it performs. The *algorithmic level* describes the representations and algorithms that system uses to perform its tasks. The *implementational level* describes how the system is physically implemented. Marr's levels have been variously reinterpreted as semantic, syntactic, and physical, or content, form, and medium (McClamrock, 1991). Figure 9.1 shows three levels of analysis applied to an example information processing system, a simple calculator[2].



[2] The CORDIC algorithm(s) used by most calculators were initially developed by Volder (1959).

Figure 9.1 Levels of analysis and an example of their application to an example system. Calculator image from https://openclipart.org.

Without necessarily committing to any particular formulation, the idea of different levels of analysis / description is both general and powerful. A lot of scientific explanation is reductive, accounting for phenomena at one level (e.g. learning) by providing a description at a "lower" level (e.g. storage in long-term memory). Conversely, some of the most interesting phenomena are emergent, where factors at one level (e.g. the behaviour of individual students) give rise to interesting properties at a "higher" level (e.g. the complex structures and behaviours of a classroom). Each level of explanation has its uses, each affords a language and a set of concepts that is distinct from, and difficult or impossible to rephrase in, the other. Each level is self-contained but incomplete; in combination they provide a more thorough understanding. These ideas may come naturally to computing educators given that many of the complex systems that we use - programming languages, operating systems, applications - make extensive use of multiple layers of abstraction (as discussed in Chapter 3.6).

CEdR has not in general been explicit about different levels of analysis. Consider two examples of how they might apply. First, a hypothetical student appears to have learned a fact one day, but not the next. A low level explanation might be the weakening of certain synapses within a hippocampal circuit. A mid level explanation might identify a failure of recall due to insufficient consolidation of the encoded knowledge during sleep. A high level explanation might note that the student was highly anxious and not sleeping well. Absent a magic pill for selectively strengthening synapses, the mid and high level explanations suggest different kinds of intervention, from explicit consolidation of material, to sleep management techniques, to addressing the underlying anxiety.

For a second example, it is frequently observed in introductory programming courses (when compared to other courses) that a larger than usual percentage of the class fail, and a larger than usual percentage achieve high grades (this is discussed at length in Chapter 3.1). Historically one explanation for this phenomenon has been that some individuals simply have an aptitude for programming and some do not - that programmers are "born and not made". The appeal to an innate capacity ("nature" over "nurture") sets this explanation in the high level framework of evolution. Consideration at this level makes clear the claim's implausibility, as the idea that evolutionary forces can possibly have selected for programming (or some adaptive programming-like analogue) is highly problematic at best. An alternative account attempts to explain the pattern of outcomes using the widely accepted educational observation that we learn "at the edges of what we know". If programming language concepts have unusually structured and interconnected "edges" then early success or failure of learning will lead to momentum towards successful or unsuccessful outcomes over all. Recognising that this explanation is framed in specific lower level mechanisms of human learning leads to hypotheses about the nature of the process, and testable predictions about programming languages as a domain of knowledge and the crucial importance of the early stages of learning to program (see Robins (2010) summarised in Chapter 3.1).

In short, recognising and making more explicit use of different levels of analysis may be of some benefit to CEdR. The remaining sections of this chapter attempt to summarise research pertaining to human cognition and learning. If at times the material seems to be too "low level" to be of educational use, consider both the popularity of reductive explanations, and that we do not know at what level of analysis the most interesting or effective account of a given phenomenon might be found.

3 Perception and attention

Perception is the process by which we as cognitive agents receive, register, and process stimuli from the external world (and from our own bodies). We begin with this low-level process to inform the higher-level process of attention. In perception, the physical or chemical stimulation of sense organs is translated into patterns of neural firing which are conducted by the nervous system to the brain. The classical senses are vision, hearing, touch, taste and smell. We have far more than five senses, though, including proprioception (our sense of the positions of parts of our body), temperature, balance, vibration, different internal chemoreceptors, and pain.

Each of the senses is a field of study in itself, for overviews see for example Mather (2016) or Goldstein and Brockmole (2016). In educational terms they suggest a focus on good ergonomics (Smith, 2007) and human computer interface design, and attention to the obvious issues raised by dealing with sensory impairment and equity of access. While the working of individual senses has generally been regarded as too early a stage of processing to be of significant interest in the educational literature, there has been some focus on "multisensory learning" e.g. Shams and Seitz (2008), see Kátai, Juhász and Adorjáni (2008) for a CEd example.

Sensory processes are not a passive "bottom-up" delivery of inputs, they are active processes incorporating "top-down" information from higher level systems. These top-down influences are evident in the kinds of artefacts of processing that cause sensory illusions (particularly visual illusions), in perceptual effects such as grouping and continuity ("Gestalt laws"), in the influence of existing knowledge or current state (e.g. motivation or expectation) on what is perceived, in the use of attention (including consciously mediated attentional focus) to select specific stimuli, and in the subjective experience of a consistent external world which emerges from complex and ever-changing inputs. Top-down effects are discussed in for example Gregory (1997), Engel, Fries and Singer (2001), Gilbert and Sigman (2007), see also the interesting debate presented in Firestone and Scholl (2016) and the subsequent commentary. An example in CEdR can be found in Hansen, Lumsdaine, and Goldstone (2013) who manipulated the white space (e.g. blank lines) in coding examples which caused confusion for both experts and novices.

Attention is the ability to select particular information from the vast and continuous array of sensory inputs. The term encompasses a range of complex phenomena, from the pre-conscious selection of specific stimuli, to overt or covert conscious selection, to the intentional

direction of high-level processing (“trains of thought”). The extent to which these are aspects of the same, or different underlying cognitive mechanisms is not yet clear. Historically a central question in research was whether sensory inputs are selected and then interpreted for meaning (called “early selection”), or whether they are all interpreted before selection (called “late selection”). In practice experimental evidence questions either model individually, and some form of compromise appears to be the case (Pashler 2016, p5), see for example the influential “feature-integration theory” (Treisman & Gelade, 1980). Attention has been variously conceptualised as: a “spotlight”, a visually based model where the spotlight consists of a focus, a margin and a fringe, e.g. Posner, Snyder and Davidson (1980); a “zoom lens”, which can trade off the size of the focus and the efficiency of processing, e.g. Eriksen and James (1986); or the allocation of limited resources such as working memory. One area of CEdR which studies attention within computing education involves using eye-tracking, e.g. Busjahn *et al.* (2014), Klaus-Dieter (2018).

In educational terms the concepts of attention, working memory and cognitive load are closely related, and all are discussed further below. Beyond the perceptual and core cognitive mechanisms summarised here, “attention” also has a behavioural and clinical interpretation in the guise of attention deficit disorders. These have significant educational and practical impacts (see e.g. Morris, Begel and Wiedermann (2015) on challenges within the technical workforce), but are beyond the scope of this review.

4 Memory and learning

Our memories are central to how we understand both the world and ourselves as individuals. Popularly, a memory is some item of information that we are able to recall to mind, but within cognitive science memory is a general term for a variety of mechanisms, including the *encoding*, *storage* and *retrieval* of information. The foundations of current views in memory were set out by Atkinson and Shiffrin (1968), who proposed a distinction between different kinds of memory: a “sensory register” (now called sensory memory), a “short-term store” (short-term or working memory), and a long-term store (long-term memory).

The material presented in this section is widely agreed upon, and rather than referencing each claim the reader is referred to some of the classic and contemporary sources, such as Norman (1970), Baddeley (1999), Tulving and Craik (2000), Baddeley, Eysenck and Anderson (2015) or Radvansky (2016).

4.1 Sensory, short-term and working memory

Sensory memory (SM) refers to the automatic and brief retention of sensory phenomena after the stimulus itself is gone. It is usually studied in the visual modality (iconic memory), but also includes hearing (echoic memory) and touch (haptic memory). In typical experiments humans are able, for a short time after exposure, to recall some details after an image is briefly seen

(less than 1 second) and then removed. As far as we are aware SM has not been explored in the context of CEdR.

Short-term memory (STM) refers to the capacity to actively hold a small amount of information in conscious focus for a short period of time (Revlin (2012) suggests up to 18 seconds). The retention period can be extended with active strategies such as conscious rehearsal.

A "small amount" of information is generally held to be 4 ± 1 items (Cowan, 2001), refining the original famous estimate of 7 ± 2 items by Miller (1956) ("Miller's Law" or the "magical number" of memory). However this estimate is somewhat complicated by "chunking", the fact that an item can contain other items. For example humans are generally able to recall a list of four single-digit numbers (e.g. 3, 7, 2, 9), but they can also recall four multi-digit numbers (72, 123, 18, 446) which between them contain many more digits. The judicious use of chunking (grouping individual pieces of information into a meaningful whole) is central to the efficient use of STM and it is a strategy which can be taught and learned (see Chapter 3.16 for further discussion of chunking).

The term *working memory* has been, and often still is, used as a synonym for STM. However most researchers accept the distinction proposed by Baddeley and Hitch (1974), that STM involves the simple retention of information, while working memory involves additional systems for manipulating this information so as to perform such activities as comprehension, problem solving, reasoning and learning. (The systems proposed by Baddeley and Hitch include a central executive, a phonological loop, an episodic buffer, and a visuo-spatial sketchpad.)

The capacity of working memory is of significant interest - why can we only concentrate on a few things at a time? Suggestions include limitations that arise from the competition for some underlying cognitive resource such as attention, or that active memory items decay or interfere with each other. It is widely accepted that the concepts of attention (the ability to selectively process information) and working memory (the ability to retain information in a useable state) are separate but closely linked (Fougnie, 2008), and that mechanisms such as attentional capture (implicitly or explicitly attending to a specific stimulus) provide inputs to working memory. Deco and Rolls (2005) describe a unifying theory of attention, short-term memory, and action selection.

Programming tasks generally involve working with a lot of information relating to the current state of the data represented and the processes being executed, the overall design and goals of the program, and also the language and tools being used. In order to accomplish this with a limited working memory programmers must make extensive use of chunking and organising knowledge structures such as schemata / plans (see below), and they must hold only parts of the overall task / program design in working memory at any given time. We know very little about how this is accomplished, or what individual variations in capacity or strategy exist. Also relevant in this context is the concept of *cognitive load* discussed below.

4.2 Long-term memory

Both sensory memory and short-term memory have a limited capacity and duration. *Long-term memory (LTM)* is the encoding, storage and retrieval of large amounts of information for significant periods of time – up to years or a lifetime. At a low level of analysis, learning is the formation of memories through the processes of encoding, storage and retrieval of information.

The *encoding* of a memory is the process by which the brain alters its structure so as to create a lasting representation of some currently perceived or attended to stimulus or item of information. Different brain regions are involved in the initial stages of encoding for different senses, but creating representations of different kinds of information (in concrete terms some pattern of neural activation) appears to depend on a common mechanism (changes to the strengths of the synaptic connections between neurons). While the hippocampus and associated structures are involved in the encoding process, the resulting representations are not (in most cases) strongly tied to any particular location in the brain (Tonegawa *et al.*, 2015).

The basic mechanism of encoding was first recognised by Hebb (1949), who proposed that a synaptic connection between two neurons is strengthened when the neurons are active at the same time. While the details are complex (depending on issues of timing and recent activity), Hebb was essentially correct, synaptic plasticity is recognised as the fundamental neural process underlying memory formation (Takeuchi, Duszkiewicz & Morris, 2014). The resulting physical trace of a memory is called an engram: “If the inputs to a system cause the same pattern of activity to occur repeatedly, the set of active elements constituting that pattern will become increasingly strongly interassociated. … We may call a learned (auto-associated) pattern an engram” Allport (1985).

Many issues can affect the success or failure of memory encoding. These include environmental factors (distractors, stressors, context), cognitive and emotional state, and the “predicted reward” associated with the stimulus. Significant events, especially those involving emotional content, appear to be strongly encoded. Encoding typically omits much of the richness of a stimulus / sensory experience, and appears to primarily represent semantic content (meaning) (Baddeley, 1966). Consistent with this observation, one of the most reliable results in the memory research literature is that encoding can be improved by “elaboration”: interacting with the stimulus at the time of encoding, processing it in meaningful ways, and in particular relating it to already existing long-term memories. This finding has had a huge impact on education, where much of teaching practice is about timing and scaffolding the introduction of content appropriately, and engaging learners actively with the new material.

Consolidation is a mechanism (variously considered either separate from or part of either encoding or storage) by which initial encoding of a memory is strengthened. Processes that occur during sleep are important to consolidation (Stickgold, 2005). These appear to involve the re-activation of recently learned information, and (particularly for novel semantic information) a transfer from initial storage in the hippocampus (and associated structures) to the cortex, over periods ranging from days to years. Good sleep is important to good learning.

If some memory (pattern of activation) has been encoded and some time later can be retrieved, then in between these times the memory is in some sense stored. Note that what is stored is not the pattern itself, but changes to synaptic connection strengths that allow the pattern to be recreated. By analogy the process of remembering a piece of music is more akin to creating a musical score than it is to making a recording.

The *storage* of memories is not a passive or a perfect system. The synaptic encoding of memories can degrade over time, or change as a result of new experiences or new memory formation. It is not currently clear exactly what mechanisms are involved in the maintenance of (the encodings of) memories, but once again processes that occur during sleep appear to play a role, particularly in protecting existing memories during new learning (Robins, 1996). One mechanism known to be important is re-consolidation. Every time a memory is retrieved, it appears to be (to a certain extent) re-encoded. Significant experimental evidence shows that memories can change over time, often in accordance with various desires, expectations or schemata (as discussed below), and the drift that occurs with multiple re-consolidations appears to be part of this process. Our memories change over time.

The *retrieval* of a memory is the recreation of the pattern of activation (or similar) that encoded the original stimulus / item. It is an active process, reconstructing the pattern from other inputs ("cues") and the encoding synapses. Two kinds of retrieval are distinguished: *recognition*, where the original stimulus is present again (and thus in effect serves as its own cue), and *recall*, in which the original stimulus is absent (other factors serve as cues). Recognition appears to be automatic, if a stimulus matches a stored memory we experience a sense of familiarity / "match". Recall can take different forms (free or explicitly cued) and appears to be a more complex process, which can involve retrieving different candidate memories and evaluating their match to the cues. Recall is improved if the context in which it is attempted (environmental, emotional or physiological) is similar to the context in which it was encoded (Tulving & Thomson, 1973). For example, information learned in a specific room will be better recalled in the same room, information learned while under the influence of alcohol will be better recalled in the same state. This is relevant to the discussion of transfer below.

At a level of analysis above these underlying mechanisms, learning can be seen as a process of forming associations between stimuli and / or behaviours. *Associationism* featured heavily in early animal learning research, such as Thorndike's "Law of Effect" (Thorndike, 1911), and it underlies the behaviourist theories of learning described below. But associationism has developed as a broad and flexible framework which contributes at different levels to various theories of thought and learning (Mandelbaum, 2017), including the learning algorithms of artificial neural networks (Hinton & Anderson, 1989).

Evidence suggests that more information is stored in the brain than can be successfully recalled, and that the phenomenon of forgetting is largely a failure of recall (Tulving, 1974). This has lead to the exploration of techniques (such as hypnotism) to assist recall, and to a popular focus on "recovered memories" which has unfortunately proved to be of very mixed validity (yes,

it is possible to assist recall, but it is also possible to "lead" it). Much of what we think we recall is inaccurate, especially as the time since the original encoding increases, and especially because recall is a reconstructive process which incorporates current cues. This is well known in a legal context where the reliability of eyewitness testimony is higher for statements taken soon after an event, and where witnesses can be "led" by the phrasing of questions.

To relate these mechanisms to a CEd context, consider for example two concerning phenomena observed in novice programmers. Firstly, the prevalence of "fragile" knowledge, i.e. knowledge which appears to be missing (forgotten), inert (learned but not used), or misplaced (learned but used inappropriately). Second, student performance which is significantly worse than expected after completion of a course. (Both of these issues are discussed in Chapter 3.1.) We would respond to these issues in different ways if they were shown to arise from partial or complete failures of attention, vs. failures of encoding, vs. failures of recall.

4.3 Kinds of long-term memory

Many different kinds of information are represented in LTM. *Declarative* (or *explicit*) memory is memory for facts or events which can be consciously recalled / "declared". It is sometimes described as "knowing that", e.g. knowing that the capital of France is Paris or knowing the syntax for a specific statement in a specific programming language. *Procedural* (or *implicit*) memory is memory for skills or motor tasks (movements) which are not consciously recalled but can be performed. It is sometimes described as "knowing how", e.g. knowing how to ride a bike or how to debug a program. Declarative memories are based on information that is consciously experienced and can be formed after a single exposure, whereas procedural memories typically require repetition / practice.

Declarative memory can be subdivided into episodic memory and semantic. *Episodic* memory is memory for events, such as a recent holiday. It is personal and roughly serial, we can locate specific episodes within the sequence of personal experiences that make up our lives. Episodic memory, and in particular *spatial* memory (relating to spatial positioning), depends on the hippocampus, and appears to be consolidated and transferred to the neocortex during sleep. In CEdR, an example of episodic memory may be when you first learned how to compile a program, and spatial memory can apply to where in a program a specific piece of code exists.

In contrast, *semantic* memory is memory for meanings, concepts or facts, such as the two times table. Semantic memory must be derived from specific learning episodes, but it is abstract and can be independent of direct personal experience, e.g. we can learn by being told, by reading, or from other indirect sources. An example of semantic memory in programming may be the behaviour of specific programming statements.

Our episodic memories generally transition to semantic memories over time, losing detail and becoming more abstract. (Most of us have things that we can "remember" about our childhood as facts, but no longer as personal experiences). Both episodic and semantic memories are

subject to error and mismatch. Episodic memories are necessarily subjective (two people may remember the same event differently), and can change over time as described above. Semantic information must be somewhat common between people, but people also construct the world differently (we can all agree on the spelling of “politician”, but not on whether a particular politician is good or bad).

While most memory is necessarily retrospective, *prospective* memory is “remembering to remember” / planning for future action (“I must remember to pick up some chocolate on the way home”). Schacter, Addis and Buckner (2007) suggest the concept of the "prospective brain", that we can recombine elements of retrospective memories to imagine / predict / simulate future events, and that humans can thereby engage in “mental time travel” (Tulving, 1985). Memory, planning, and mental models of future action, are all closely related.

Programming must involve a considerable demand on both memory and prospective memory. The programmer must “remember to remember” to close structures at multiple levels, from local constructs such as loops, to the functionality of organising structures such as classes, to overall program designs / patterns. In general terms, planning and problem solving requires both retrospective and prospective memory, as possible solutions to goals and subgoals are explored and discarded or adopted.

4.4 Memory structures and representations

How is information organised in memory? These issues are explored in the context of a higher level of analysis, but are assumed to rest on the lower mechanisms just described.

The most fundamental structure is the *category*, a collection of items that we recognise as being instances of the “same thing”, such as a chair, a tree, or a person. “There is nothing more basic to thought and language than our sense of similarity; our sorting of things into kinds” (Quine, 1969). There is strong evidence that there exist “basic level categories” characterised by obvious similarities, that humans learn first and react to most strongly (Rosch *et al.*, 1976). For example “dog” is a basic level category, with superordinate categories (such as “animal”) and subordinate categories (such as “Labrador”) emerging later in development. The hierarchical organisation of categories is sometimes described as a *type hierarchy*, with instances of a type being *tokens*.

The question of how categories are represented in the brain is a matter of very active debate. The *probabilistic* approach suggests that categories are represented by some measure of the central tendencies of instances, such as a set of characteristic features or dimensions, or an instance-like “prototype”. The prototypical dog doesn’t exist, but we all know a lot about it. The *exemplar* approach proposes that categories are encoded not by a single representation, but by the separate representations of salient instances. Individual instances may be more significant than an abstract average, for example our earth is a very influential instance of the category “planet”. Both of these approaches can account for graded category membership (a sparrow is

a better example of a bird than is a penguin) and fuzzy category boundaries (when does a stream become a river?). Experimental evidence supports both approaches, suggesting that we use a mixture of strategies (Medin, Altom & Murphy, 1984; Minda & Smith, 2002, Hampton, 2016).

One construct that is problematic for standard models of categorisation is the “ad hoc” category (Barsalou, 1983). These are the “categories” that can be created by an arbitrary specification. For example consider the category of “things you would rescue if your house was burning down”. It has a similar structure to traditional categories, including graded membership (good and bad instances) and prototypical members (rescue other people!), but it appears to emerge from an active process, difficult to account for in terms of the structure of long-term memory.

Speculatively, it may be interesting to explore the relationship between programming constructs and cognitive categories. Data types define categories of data which can be treated in the same way, we have understandings and expectations about both the type/category and specific tokens/examples. Given that object oriented programming was developed as a tool to give us "natural" ways of thinking about tasks it is not surprising that object types are hierarchical, and can be prototype based. Where new data types/classes are created these could be seen as examples of ad hoc categories.

More specialised kinds of memory structures have been described, within cognitive psychology and artificial intelligence, by the related terms schema, frame, script or plan (Schank & Abelson, 1975, Whitney, 2001; Schank & Abelson, 2013). A *schema* is a unit of declarative knowledge which organises items of information and the semantic relationships between them. Examples include schemata for objects such as triangle or horse, for scientific constructs such as a specific logic or the periodic table, for archetypes / social roles such as teacher or parent, or for belief systems such as a religion or “worldview”. They are “generic representations” of “preconceived knowledge” which structure our perception and cognition both individually and collectively (a particularly salient example being social schemata such as racial or political stereotypes). Schemata also influence memories over time, as “typical” information not present at the time of encoding is added during episodes of recall and re-consolidation. A *frame* is a structure consisting of “slots” and “fillers”, e.g. the “teacher” frame might consist of slots representing subject and school level, with default values such as “maths” and “secondary”. A *script* or a *plan* is an “action oriented” schema for some sequence of actions or events, e.g. a birthday party script or a restaurant script. These can be shown to influence the recall of descriptions by adding common / script details (e.g. adding birthday presents) which were not present in the memorised material.

Schemata / plans were adopted by the CEdR community during the 1970s to 90s, and featured heavily in research on the “psychology of programming” that was popular at the time. Expert programmers are characterised by a large library of useful schemata / plans which chunk and organise significant amounts of information in a form which reduces cognitive load. Many of the difficulties experienced by novice programmers arise from their lack of such organising knowledge structures. These issues are discussed in some detail in Chapter 3.1.

4.5 Memory in practice

Much of what is known about practically improving memory, such as actively engaging with and elaborating new material, has been incorporated into standard pedagogical practice and typical "study skills" advice to learners (Dunlosky *et al.*, 2013). Active strategies for remembering date back to the Greek use of the "method of loci" (in the context of rhetoric), which draws on well practiced spatial memory to organise and help remember other material at need. Today there are many proven mnemonic techniques, and many popular books, websites and other resources which explain and promote their use (e.g. Foer, 2012). We are not aware of any specific exploration of study skills or application of mnemonic techniques in CEd.

Similarly, there are many general issues that impact on learning and memory, such as health, age, diet, stress, exercise and sleep, which are well known in clinical contexts and widely discussed in popular sources. Various physical and cognitive disorders affect memory, including amnesias, agnosias, the effects of brain injury, and conditions associated with ageing such as Alzheimer's disease or dementia. It is not likely that any of these effects would be specific to our field.

5 Topics in cognition

In this section we give very brief overviews of some of the important topics in cognition. In the remaining sections of the chapter we explore in more detail two topics of particularly practical significance to educators, the transfer of learning and cognitive load.

5.1 Cognitive development and language acquisition

Humans acquire knowledge, skills and capabilities on different timescales and as a result of different processes of physical, neurological and cognitive maturation / development and learning. The field of cognitive development has fundamentally shaped educational theory and practice. It is also of increasing relevance to CEd, as computing makes its way into many early school curricula.

Jean Piaget was the first to formalise the observation that growing children pass through roughly predictable stages, effectively founding the field of cognitive development with his *developmental stage theory* (Piaget, 1964, 1971a, 1971b). Piaget's early stages introduced such concepts as object permanence and conservation (of volume, number or mass). The later stages are defined largely in terms of the acquisition of logical capacities such as transitive inference. The fundamental unit of knowledge in the theory is the *schema*, a cohesive, repeatable "action sequence" organised around a "core meaning" (Piaget, 1952). Learning is seen as involving *assimilation*, the integration of new information into schemata, and

accommodation, the process of adjusting schemata (when existing ones do not suffice). In a large scale meta-analysis of factors influencing educational outcomes (Hattie, 2009; 2012) "Piagetian level" of development was identified as one of the "super effects", the higher a student's assessed level the better their outcomes (Killian, 2016).

Piaget's framework had a huge impact on curriculum design and educational practice internationally. It introduced concepts which are now taken for granted, such as the active engagement of the child with learning, and was one of the major influences on the now dominant educational philosophy of *constructivism*. His impact on education and the field of cognitive development is difficult to overstate (Flavell, 1996; Scholnick *et al.*, 1999).

Aspects of Piaget's original theory, however, were problematic, particularly the observed variation between individuals, and the lack of explanation for the "miraculous" transition between stages (Feldman, 2004). It is not clear what the relationship is between individual differences in development and individual differences in IQ, and what exactly is measured by the different tests for these attributes (Sternberg, 1982; Cianciolo & Sternberg, 2004). Attempts to address such problems led to various "neo-Piagetian" theories, these are discussed in the context of novice programmers in Chapter 3.1.

Other influential ideas in development arose from the work of early psychologist Lev Vygotsky, for an overview see Rieber and Robinson (2004). Vygotsky noted that there was a distance between children's developmental level as determined by their independent performance, and their potential level when assisted by a teacher or in a social context. Vygotsky called this space, within which children are capable of learning and making progress, the *Zone of Proximal Development* (ZPD), suggesting that this was the most effective level to target instruction and assistance. The ZPD has become a standard and influential concept in both cognitive development and education, see for example Wertsch (1984), Chaikin (2003), and discussion in Chapters 2.5 and 3.13.

The term *scaffolding* (Wood, Bruner & Ross, 1976) refers to the support given to the learner within their ZPD. Such support includes direct instruction or advice, resources, modelling, and well designed tasks, and it is reduced / withdrawn as the learner becomes more independent (see Chapter 2.5). In contrast, *bootstrapping* (Bereiter, 1985) describes the process of learners progressing independently, without the assistance of others (Bereiter explores ten possible mechanisms which might support this). For overviews of further topics in cognitive development and links with other disciplines, see for example Fischer (1980), Wertsh and Tulviste (1990), Flavell (1992) and Bandura (1993).

One major interdisciplinary topic spanning development, linguistics and education, is the acquisition of language (McNeill, 1970). It is widely claimed that there are maturational constraints on language learning (Newport, 1990), that there are critical periods in the process (Lenneberg, 1967; Singleton & Ryan, 2004), and that children learn languages easily based on procedural memory, while adults learn via a more explicit process based on declarative memory (Ferman & Karni, 2012). While we are not aware of any suggestion that children learn

programming languages more easily than adults, the recent addition of computing in to the school curriculum highlights the need to explore children's learning processes, and the way they may differ from adults in this domain.

An examination of the vast field of language is beyond the scope of this chapter, for an overview see Crystal (2010). In the context of CEd, Bonar and Soloway (1989) explore preconceptions based on natural language as a source of misconceptions about programming (for more see Chapter 3.16), and there has been some discussion of parallels between learning a programming language and the acquisition of a second (human) language (Baldwin & Macredie, 1999; Pandža, 2016). Ullman (2001, 2004) proposed a model of language learning where lexical forms (words) are stored in declarative "rote memory", and grammar is a set of rule-like mental operations that is dependent upon procedural memory. Do the lexical elements of programming languages and the knowledge of how to combine and use them exhibit a similar dichotomy?

5.2 Behaviourist accounts of learning

Behaviorism was the dominant school of thought within psychology during the first half of the 20th century. *Methodological behaviourism* (Watson, 1913; 1930) holds that the only scientifically valid object of study is observable behaviour, any consideration of subjective, phenomenological or mental states or processes is excluded. Behaviour in this framework is seen to arise as a result of environmental stimuli interacting with an individual's history of reinforcement or punishment and/or current drives (e.g. hunger). *Radical behaviourism* (Skinner, 1938; 1953) acknowledges the existence and valid study of "private events" (e.g. thoughts, feelings, mental states) but holds that environmental factors control these just as they control observable behaviours. Internal states cannot explain behaviour (such attempts circularly presuppose their own explanation), instead they should be understood and translated into behavioural terms.

Behaviourism was rooted in earlier work on animal learning, which it developed and applied to animals and humans in two main theoretical and practical directions. *Classical conditioning* involves the association of innate reflexes with new stimuli (the famous example being Pavlov's dog, taught to salivate at the sound of a bell). *Operant conditioning* is the modification of behaviour using *reward* (stimuli which increase its frequency) or *punishment* (stimuli which decrease its frequency). Reward or punishment may be *positive* (added to the environment) or *negative* (removed from the environment). Complex behaviours may be *shaped* by the careful reward of sequences of antecedent or partial behaviours. Rewards or punishments on various *schedules* can result in behaviour with mixed patterns of frequency and robustness. For a review and modern synthesis see Bouton (2016).

Despite the fact that the underlying theory fell out of favour, behaviourism achieved major and practical insights into both animal and human learning that are still widely employed today. Classical conditioning lead to what are now called behavioural therapies (such as aversion

therapy, systematic desensitisation and flooding). Similarly operant conditioning led to applied behaviour analysis ("behaviour modification"), the practical modification of behaviour in fields such as behavioural interventions for children, criminology, animal training, addiction and dependence, health and exercise, and more. Reinforcement (applied under various schedules) is also widely used in the gambling and gaming industries, including video and online games – it is no accident that they are "addictive". Reinforcement based management of behaviour in general is also important in economics, workplace and business management, and education (McSweeney & Murphy, 2014), and has made its way into popular culture with the understanding that we should "reinforce good behaviour". Concepts from this tradition have also influenced reinforcement learning, a major topic in the context of machine learning / artificial intelligence (Sutton & Barto, 1998).

In the context of education *behaviour analysis* is a field of interest (Cooper, 1982; Twyman, 2014), and behavioural tools are an important aspect of classroom management (Emmer & Stough, 2001; Woollard, 2010). We are not aware of any significant exploration of behaviour based methods in CEd or CEdR so far, but suggest that there are two emerging topics where they could be relevant. The first of these is the "gamification" of learning (Kapp, 2012), especially as this might be applied to the teaching and learning of programming and computing topics. An understanding of reinforcement schedules and behavioural techniques will help to maximise the efficacy of game-based teaching and learning.

A second area of potential application is suggested by the field of behaviour informatics / behaviour computing (Cao, 2010; Cao & Philip, 2012). This field encompasses the extraction of information about patterns of behaviour from large volumes of data (e.g. financial transactions, system use, social media) and its use for understanding, modelling and predicting future behaviour (e.g. risk analysis, marketing design, load prediction). As CEd is now adopting the use of learning analytics (Chapter 3.12) and large bodies of data collected from instrumented compilers (Chapter 3.10), there is an opportunity to explore these related tools and ideas. What do our datasets tell us about events that make certain behaviours more or less probable? Can we add reinforcement to our tools to shape appropriate behaviours?

5.3 Cognitivist accounts of learning

Arising from the late 1950s "cognitive revolution" and superseding behaviourism as the dominant paradigm within psychology, cognitivism invited the study of internal / mental states and processes (see further discussion in Chapter 2.7). The underlying assumptions view the brain as a processor of information, and thought as a form of computation (Lindsay & Norman, 1977; Glass, Holyoak & Santa, 1979).

The account of memory and learning set out in Section 4 is cognitivist. It describes functional organisation and internal processes such as attention, and working and long-term memory. It describes learning / memory formation (at a low level of analysis), and (at a higher level) knowledge structures such as cognitive categories and schemata. It explores aspects of

internal representation such as the prototype vs. exemplar accounts of categories. This is the context within with CEdR has developed, particularly with the focus in the 1970s to 90s on the "psychology of programming". Aspects of learning theory which have been particularly influential within CEdR include the central importance of organised knowledge structures such as schemata / plans, and the application of various learning taxonomies (such as the Bloom (Bloom *et al.*, 1956) and SOLO (Biggs & Collis, 1982) taxonomies), as described in Chapter 3.1.

Work within the cognitivist tradition has explored a range of specific forms of knowledge representation and/or processing, such as semantic networks, frames, decision trees, production rules, blackboard models, and more (Brachman & Levesque, 1985; Sowa, 2000). From the many formal models of cognitive process, two particularly ambitious examples stand out. The Adaptive Control of Thought (ACT / ACT-R) model (in various iterations) of John Anderson (Anderson, 1976; 1982) was based on an associative network model of LTM and production rules representing procedural memory. It encompassed problem solving and a model of learning as the compilation of declarative into procedural representations. The Soar Cognitive Architecture (in various iterations) of Rosenbloom, Laird, Newell and colleagues (Rosenbloom *et al.*, 1991; Laird & Rosenbloom, 1996) is based on representations of search spaces and operators which can be applied to them. It encompasses different kinds of memory and representation, problem solving, and learning as a process of reinforcement and the chunking / organisation of the rules that trigger operators. Soar remains an ongoing research programme (Soar, 2018).

In the CEdR literature, Rist (1995) proposes an example of a comprehensive formal model of a complex topic, the process of program generation. Knowledge is represented using indexed nodes (encoding actions) in memory (working, episodic, and semantic). A program is built by starting with a search cue such as <find, average, rainfall>, and retrieving and expanding any matching nodes. Outputs are called plans, and common / useful plans are assumed to be stored as schema–like knowledge structures. Experts can retrieve relevant plans from memory, but novices must typically create plans through a process of "focal expansion".

In an exploration of novice programmers, Robins (2010) appeals to the educational principle that new learning builds on existing knowledge – we learn at the edges of what we know (e.g. the concepts of zone of proximal development, scaffolding and bootstrapping discussed above). For some knowledge representations learning can be described as, for example, "a series of local repairs of a knowledge structure" involving mechanisms such as adding and adjusting links (Chi & Ohlsson, 2005). Robins suggests that different domains of knowledge have edges and interconnections of different complexity, and that in the case of densely connected domains (such as programming languages) success or failure in the integration of new knowledge can have a compounding "momentum" that significantly affects learning outcomes.

To sum up, of the two major learning frameworks to succeed behaviourism, cognitivism as discussed above grew out of psychology and artificial intelligence. Constructivism (as widely discussed elsewhere in the Handbook, see Chapters 1.2, 2.5, 2.7, 2.8, 3.4, 3.13 and 3.18) grew

out of education and child development. See Ertmer and Newby (1993), and Cooper (1993) for reviews and comparisons of all three frameworks.

5.4 Reasoning and problem solving

Reasoning has been variously defined as including thinking, understanding, deciding, making sense, evaluating, problem solving, changing beliefs, and other abstract (high level) descriptions of aspects of cognitive function. For an overview of this broad topic see Holyoak and Morrison (2005).

Early attempts to formalise reasoning gave rise to logic – rules and principles for drawing valid conclusions. *Propositional* logic (propositions that can be true or false, logical operators) was set out by George Boole (1815 - 1864, now immortalised as a data type). From this foundation a succession of theorists developed further logical systems, and formalised concepts such as syntax, semantics, proof, and forms of reasoning such as deduction and induction (see e.g, Enderton, 2001). Logic underlies the theory and practice of computation, early work in Artificial Intelligence, approaches to design and verification, forms of economic modelling, and more.

Humans can use logic. It defines some kind of ideal, but human reasoning does not strictly adhere to logic. We are prone to formal fallacies (e.g. affirming a disjunct, denying the antecedent, base rate fallacy…), practical fallacies (e.g. gambler's fallacy, circular reasoning, false equivalence…), and a bewildering array of cognitive biases (e.g. confirmation bias, framing bias, stereotyping…), see Pohl (2004) for an overview. Further exploring the impact of known fallacies or biases in the context of specific computing topics seems like a productive area for future research. Fallacies so far observed within CEd (mostly relating to programming) have been termed misconceptions, as described in Chapter 3.16.

If not logic, what? *Human Problem Solving* (Newell & Simon, 1972) sparked a wave of cognitivist research into reasoning. The authors explore problem solving within the constraints imposed by the "human information processing system" (serial processing, a small short-term memory, a large long-term memory with fast retrieval but slow storage), and stress the importance of "simple schemes of heuristic search". The topic of heuristics was further developed by Tversky and Kahneman (1974). Noting that most reasoning takes place in the context of incomplete / uncertain information, the authors review and present evidence relating to three heuristics which pervasively influence human cognition: availability (the ease with which relevant information can be recalled); representativeness (the impact of similarity and prototypes); and anchoring (the impact of initially stated information). A subsequent programme of research by these and other authors identified and categorised further heuristics. These do not appear to have received much attention in CEdR. Is novice reasoning about programming influenced by common cognitive heuristics? If so, how, and how do we take account of this as educators?

Another framework used to explore reasoning is mental models (Gentner & Stevens, 1983; Johnson-Laird, 1983; Gentner, 2002). These are internal models of how some aspect of the world works, an iconic representation of selected aspects of external objects and systems. Mental models have predictive power, they can be used to understand the observed behaviour of the world and reason about future behaviour. The use of mental models generally, and of a particular kind of mental model (the "notional machine"), has been widely adopted in the CEdR literature as described in Chapter 3.1.

Problem solving is a pervasive and practical version of the topic of reasoning, discussed in contexts from solving mathematical puzzles, to managing businesses, to interpersonal relationships. The range of formalised problem solving strategies includes for example: divide and conquer (breaking a problem down into smaller, solvable problems); analogy (adapting the solution to a structurally similar problem); and means-ends analysis (choosing an action at each step to move towards the goal). Within the cognitivist framework problem solving is usually construed as a process of search within some representation of the problem space. For more on this, and a range of perspectives on problem solving, see Schoenfeld (1985), Novice and Bassok (2005), Wang and Chiew (2010) and Robertson (2016).

Individuals vary in their capacity to solve problems and perform tasks, which is one of the topics explored under the general heading of intelligence, as formalised in various intelligence quotient (IQ) measures. The literature relating to intelligence is diverse, with debate as to whether it should be thought of as single general factor "*g*", a few major factors (such as verbal and spatial intelligence), or multiple factors (such as the logical, linguistic, spatial, musical, kinesthetic, naturalist, intrapersonal and interpersonal intelligences proposed by Gardner (1993)). One of the most general results, however, is that performance on one standard psychometric test is highly predictive of performance on other such tests, a phenomenon known as "the positive manifold" (Jensen, 1998) and sometimes used as an argument for the existence of *g*. The importance of intelligence in programming and CEd is discussed by Pea and Kurland (1984) and reviewed in Robins (2010).

In CEd practice, problem solving is usually introduced, along with algorithms, in a first programming course ("CS1"). Courses vary in the prominence they give the topic, and its order in the curriculum, but many CS1 courses are explicitly "problem solving based", as described in the ACM curriculum guidelines (ACM/IEEE-CS, 2013). An increasing focus on problem solving in CEd was noted by Caspersen and Bennedsen (2007), with most (but not all) studies suggesting improved outcomes. Hazzan, Lapidot and Ragonis (2014) set out practical guidelines for teaching problem solving. Many studies relating to problem solving in the CEdR literature are reviewed in Chapter 3.1.

One of the most important lessons from the study of problem solving is that it can be a highly contextual process. A widely discussed illustration of this effect is the reported ability of children to perform mathematical operations in the context of purchases (e.g. "making change"), but performing significantly worse on the same calculations when presented as "maths problems".

The classic study on “street maths” is Carraher, Carraher and Schliemann (1985), the many follow-up studies include current work in India by Banerjee *et al.* (2017). Further important examples of the contextualisation of problem solving are discussed in Chapter 3.17. This is the issue underlying the importance of the topic of transfer in teaching and learning, as discussed in the next section.

6 Transfer

Because of its practical significance to educators, we have chosen to focus in some detail on the topic of transfer, which is a major theme within the learning literature. This section will illustrate how some of the lower-level processes, such as attention and memory, are applied to higher-level phenomena, such as problem solving and learning.

6.1 What is transfer?

For the purposes of CEd, we define transfer as a student’s ability to transfer the knowledge that they’ve learned and apply it to solving new problems. For example, if a student learns to write a program to find the average of a waiter’s tips, then they should be able to write a program that finds the average score on a test. Instructors and content creators tend to think that transfer will come naturally to learners, but in reality, learners, especially those who have little knowledge of the domain, struggle to apply what they (tenuously) know to novel problems. Trouble with transfer is best documented in computing by performance on Soloway’s Rainfall Problem. Soloway’s Rainfall Problem has been given to students at the end of introductory programming courses to test their knowledge on several concepts. Students commonly perform poorly on this problem even though their instructors think that it is a straightforward application of the concepts they have learned throughout the semester (Guzdial, 2015).

Before we discuss what makes transfer so difficult for learners and how to promote it in our students, let’s consider the various types of transfer. The type of transfer that we want to enable is spontaneous transfer. In spontaneous transfer, the student recognises the similarities between what they already know and the knowledge required to solve the new problem without help or guidance (Bassok & Holyoak, 1989). When spontaneous transfer doesn’t occur, students can often be guided to transfer by pointing out the similarities between old and new problems or asking the student to find similarities (Gick & Holyoak, 1980). Because instructors will not always be around to provide guidance, spontaneous transfer is our ultimate goal. Spontaneous transfer is related to transfer distance.

6.1.1 Types of transfer

Barnett and Ceci (2002) provide a comprehensive taxonomy for transfer distance. They created the taxonomy to identify dimensions that can be used to describe different types of transfer in a more sophisticated manner than only “near” or “far,” which is commonly how it had been described in literature. The first dimension of transfer that Barnett and Ceci discuss, which is by

far the most common dimension that people explore, is knowledge domain. For this reason, it is discussed in more depth than the others.

1. *Knowledge domain transfer* is transfer within or between different disciplines.
    - Near transfer - transfer within a discipline
        - Isomorphic transfer - the problem-solving context (i.e., cover story) between problems is the same, and only the values in the problem are different. The term *isomorphic* comes from the math field (Bassok, 1990), but it has been applied (transferred, if you will) to similar scenarios in other fields (e.g. Morrison, Margulieux, & Guzdial, 2015).
        - Contextual transfer - the problem-solving procedure between problems is the same, and the contexts of the problem are different. For example, transferring knowledge from writing a program to find the average value of a waiter's tips to writing a program to find the average rainfall is contextual transfer. The context difference might cause slight variations in the problem solution, but the general steps of the procedure are the same.
        - Procedural transfer - the type of problem is the same, and the problem-solving procedures are different. For example, transferring knowledge from writing a `while` loop to find the sum of values to writing a `while` loop to find the average of values is procedural transfer. The steps of the procedure differ based on the components of the problem (summing or averaging), but the type of problem (using a `while` loop) is the same.
    - Far transfer - transfer between disciplines
        - Several shared elements - the domains share several elements that make general problem solving procedures applicable to both. For example, learning to create algorithms in math can be applied to creating algorithms in computing, and the general problem-solving strategies used to achieve both are similar.
        - Few shared elements - the domains do not share many elements, making transfer possible at only an analogical level. For example, writing about a sequence of events in English is conceptually similar to writing a program for a sequence of events, but the procedures and knowledge required to complete each are very different.
2. *Physical context transfer* is transfer among spaces (e.g. classroom, dorm room, kitchen table at parents' house). Because information is encoded with the physical context in which it was learned, location can affect transfer.
3. *Temporal context transfer* is transfer across time (e.g. weeks, months, semesters). Because synaptic connections fade over time (i.e., ability to recall information decays), the time between solving problems can affect transfer. Time transfer is also commonly called **retention**.
4. *Functional context transfer* is transfer among purposes (e.g. authentic web development problems, academic exercise to learn concepts used in web development).
5. *Social context transfer* is transfer among group compositions (e.g. solving problems individually, solving problems in teams).

6. *Modality transfer* is transfer between medium of instruction and application at the macrolevel (e.g. auditory instructions vs. written assignment) or microlevel (e.g. instructing based on tracing, reading, or writing code vs. testing based on tracing, reading, or writing code).

6.1.3 Why is transfer so hard?

Reviewing these six dimensions should give you an idea of how contexts can change between learning to solve a problem and solving a novel problem. Any contextual differences that make it difficult for the learner to recognise similarities between problems will hinder transfer (Gick & Holyoak, 1980). The more abstract instruction is, the more easily students can transfer knowledge spontaneously (Day & Goldstone, 2012). Abstract instructions, however, are very difficult for novices to grasp because they do not have enough knowledge about the domain to connect abstract information to their existing cognitive architecture (i.e., connect to prior knowledge), making it difficult to retrieve (Jonassen, 2000). Therefore, providing surface details, i.e., cover stories, for problems is essential to teaching novices, even though those surface details can promote ineffective problem solving schemata and mental models (Eiriksdottir & Catrambone, 2011).

Surface details promote ineffective knowledge structures in long-term memory because novices have difficulty seeing past surface details to identify structural parts of problem solving procedures that apply to all similar problems (Bransford, Brown, & Cocking, 1999). Instead of mentally organising information around structural details, they create knowledge structures around surface details (Chi, Feltovich, & Glaser, 1981). For example, novice programmers are more likely to organise information around an application type, whereas more advanced programmers will organise information around an underlying algorithm (Weiser & Shertz, 1983). Therefore, novices' mental organisations of information often suppress spontaneous transfer. To fix these issues, we need to help learners identify structural parts of problem solving procedures, and enable them to connect them to a variety of problems.

6.2 How to promote transfer within a domain

The best way to help students transfer their knowledge is to teach them how to transfer their knowledge. Cognitively speaking, learning content is not the same as learning how to transfer content (Bransford *et al.*, 1999; van Merriënboer, Clark, & de Croock, 2002). Too often instructors will assume that if students see enough examples and understands the content, transfer will come automatically. Instead, instructors should tell students how to recognise similarities between novel problems and problems that they know how to solve, and they should give students practice and feedback on novel problem solving (Singley & Anderson, 1989).

6.2.1 Frameworks for helping students to see past surface details

One of the biggest problems that keeps students from successfully transferring their knowledge is that they don't recognise similarities between problems that they know how to solve and new

problems, called the inert knowledge problem (Reed, Ernst, & Banerji, 1974). In one of the classic transfer studies, Gick and Holyoak (1980) showed participants a problem in which a general was attempting to take a town, but his army was too big and vulnerable to attack to march into the town on the main road. The solution was to split the army up and converge on the town from multiple directions at once. After Gick and Holyoak taught participants to solve this problem, they asked participants to solve Duncker's radiation problem, a problem in which a doctor is attempting to treat a patient with a tumor. The tumor can be destroyed with a laser ray, but if the laser is at a high enough intensity to destroy the tumor, it will also destroy all of the healthy tissue in the laser's path. The solution is the same as the army problem, align several low intensity lasers to converge on the tumor so that the tumor is destroyed and the healthy tissue is not. Despite the similarities in problems and solutions, only 20% of participants were able to solve Duncker's radiation problem, even though they just learned the solution to the army problem. When participants were told to find analogies between problems or given hints about similarities between problems, 92% of participants successfully solved it. The takeaway from this and related studies, is that students need to be guided or told how problems are similar, until they have developed a generalised mental model or schema for problem solving that can recognise structural similarities between problems.

Helping students to identify structural similarities between problems is important in computing education because we typically rely on worked examples to help teach computing concepts. A worked example is a problem with a worked out solution that students can study before they are able to solve problems independently (Renkl, 2017). We have strong evidence that asking students to study worked examples is more effective in the early stages of learning than asking them to solve problems from abstract instructions (Renkl, 2017; Sweller, 2010). The problem with worked examples is that they show learners how to solve only one problem because examples have to include a cover story. Just like in Gick and Holyoak's (1980) army problem, students who study worked examples are able to solve novel problems better when they are guided to see past the surface details.

The subgoal learning framework has been used to help students see past surface details of worked examples. Subgoals are functional pieces that are inherent in problem solving processes (e.g. initiating a loop is a subgoal in a problem that uses loops). The worked example is visually broken down into subgoal components, then each subgoal is meaningfully labeled (Margulieux, Guzdial, & Catrambone, 2012; Morrison, Margulieux, & Guzdial, 2015). The subgoal label, a short instructional explanation of the purpose of the subgoal, is abstract and does not contain any of the surface details of the problem, meaning that it can be applied to any similar problem. In the subgoal learning framework, students are given multiple examples, each with different surface features but with the same subgoal labels, so that they can compare and contrast the problem solving steps that achieve the same subgoals across multiple examples (Margulieux *et al*., 2012). For example, in a `while` loop worked example, one of the subgoal labels could be "determine initial values," in which you would specify values before the loop begins (Morrison *et al.*, 2015). Explaining the purpose of steps in a worked example at this low level helps novices to transfer their knowledge to solving novel problems and retain the problem solving process for longer (Margulieux *et al.*, 2012; Morrison *et al.*, 2015).

Analogical encoding is another strategy that is used to help students recognise the common structure between examples (Ferguson, 1994). It is based on analogical reasoning, in which people create analogies between fundamentally similar concepts (Gick & Holyoak, 1983). Creating analogies helps students to recognise when two examples share the same problem solving procedure, promoting development of schemata (see more about schema instruction below; Kurtz, Miao, and Gentner (2001)). When students are encouraged to create analogies, their transfer to novel problems improves, whether they receive guidance on the similarities or not (Gentner, Loewenstein, & Thompson, 2003). Please note that analogical encoding is different than using analogies in instruction. In analogical encoding, students are the creators of analogies and the analogies are between two examples of the same problem solving process. When an instructor gives students an analogy, the student is not creating the analogy and the analogy is typically between two disparate examples of similar concepts: the concept that they are learning and a concept that they already know (e.g. electricity is like flowing water; Gentner, 1983).

In contrast to analogical encoding, which helps students to build schemata for themselves, schema instruction is a method of explicitly training students on schemata in a field to jumpstart their problem solving potential before they have enough experience to develop schemata themselves. As discussed earlier, a schema is a framework for solving a class of problems (Marshall, 1995), like an abstract recipe. There are two types of schema instruction (Powell, 2011). The first is schema-based instruction in which instructors teach students a schema or set of schemata and also teach students to classify problems to match a problem to the appropriate schema. Examples of schema-based instruction in programming education can be found in “How to Design Programs: An Introduction to Programming and Computing” by Felleisen *et al.* (2001). The second is schema-broadening instruction, which is more focused on transfer. In schema-broadening instruction, instructors help students to match problems that have new features (e.g. a different problem structure) to an existing schema (Powell, 2011).

While all of these approaches to improving transfer are very similar, they have some differences. They differ in the amount of direct instruction that students receive about underlying problem solving structures (e.g. subgoal labels explicitly describe the structure whereas analogical encoding asks students to identify the structure). They also differ in the source of schemata (e.g. schema instruction provides schemata and analogical encoding asks students to create them). Though the approaches have never been empirically compared, their efficacy likely depends on the content being taught, the prior knowledge of the student, and the style of the instructor. For reasons that will be discussed in the next section, more complex content or students with less prior knowledge will probably benefit from more direct instruction approaches, such as subgoal labels and schema instruction. For content with less novel information or students with significant prior knowledge, more student-driven approaches, such as analogical encoding, would probably be more effective.

6.2.2 Helping students to retain knowledge

Recognising structural similarities between problems is likely the biggest barrier to transfer, but transfer also suffers when neural connections to relevant prior knowledge decays over time, causing a lack of retention, or time transfer. A common retention problem occurs in classes that have prerequisites. Instructors often complain that their students do not have the prior knowledge that they should have from previous classes. Assuming that students did learn what they needed to in previous classes, however, they would have a hard time activating it if their neural connections have sufficiently decayed. The good news is that it is much faster to re-learn material, i.e., re-strengthen neural connections, than it is to learn it for the first time (Hansen, Umeda, & McKinney, 2002). To promote retention, instructors should focus on learning and teaching strategies that will develop strong neural connections that will take a long time to decay.

A major factor in retention is how students learn knowledge. The default teaching style in higher education is direct instruction, meaning that an instructor tells students what they need to know. Many argue that this method is more effective and less taxing on learners than constructive methods, in which students build knowledge for themselves by exploring the problem solving space (Kirschner, Sweller, & Clark, 2006). Constructed knowledge, however, often leads to better retention in the long run, when it is successful (Tobias & Duffy, 2009). Learning constructively by exploring problem spaces benefits from the generation effect. The *generation effect* states that learners are more likely to remember information when they generate it for themselves than when they are told it because the cues that they encounter while encoding information are similar to the cues that they encounter when they solve problems (Jacoby, 1978; de Winstanley, Bjork, & Bjork, 1996). Therefore, learners are more likely to activate relevant prior knowledge to solve problems if they constructed that knowledge while working on problems. Constructive learning, however, is not appropriate for all learning situations (e.g. when learners have little prior knowledge) and should be carefully conceptualised and planned before it is used (Tobias & Duffy, 2009).

When learning constructively is not a good option, consider how learners are practicing using the information that they have been told. A large body of research supports that spaced practice, in which learners solve problems across several sessions, results in better retention than massed practice, in which learners solve problems in one sitting (Donovan & Radosevich, 1999). Massed practice can be good while students are building schemata and learning to recognise similarity between different examples, but spaced practice requires that learners devote attention to other tasks and then re-focus on the problem solving procedure, resulting in better long-term retention (Donovan & Radosevich, 1999). Most spaced practice studies have intervals from minutes to weeks, but spaced practice also explains why students who take cumulative final exams retain knowledge better (though students expecting a final exam also spend more time integrating information; Szpunar, McDermott and Roediger (2007)). In related research, Trafton and Reiser (1993) found that interleaving studying examples and solving problems improves retention. In their paradigm, students studied examples, and then applied what they had learned to solving a problem before studying a new example. This interleaved approach was more effective than studying all examples at once and then solving all problems at once (Trafton & Reiser, 1993).

The last technique to improve retention that we will discuss is the timing of feedback on problem solving practice. In some cases, immediate feedback is the most effective type of feedback. Especially in language learning, which is related to syntax learning in programming, immediate feedback to fix errors is most beneficial (Kulik & Kulik, 1988). In more conceptually focused problem solving, however, slightly delaying feedback can allow learners to recognise when they have made a mistake and fix it for themselves, improving retention (Mathan & Koedinger, 2003). Related to the spaced practice effect, delaying feedback by a significant amount of time (i.e., enough time to switch tasks) gives the learner another opportunity to encode information about correctly implementing the problem solving process (called the delay-retention effect; Smith and Kimball (2010)).

6.3 Can we promote transfer between domains?

Sometimes problem solving strategies can be applied in multiple domains. For example, students' knowledge of algorithms should transfer to support problem solving in math and science. The difficulty with cross-domain transfer, however, is that it becomes very difficult for learners to recognise commonalities among problem solving contexts (Barnett & Ceci, 2002). Spontaneous transfer within a domain is already difficult; therefore, some cognitive scientists believe that spontaneous transfer across domains is very rare, at least not for novices (Anderson, Reder, & Simon, 1996; Brown, 1992; Feldon, 2007; Singley & Anderson, 1989). They argue that knowledge is embedded in the context in which it was learned (e.g. in a classroom, with an example about averaging a waiter's tips, in LISP; Anderson *et al.*, 1996; Singley and Anderson (1989)). Therefore, if you want to promote transfer across domains, then you have to include contexts from all domains in the learning of knowledge (Brown, 1992). Including multiple domain-specific sets of information, however, places unreasonable demands on learners' cognitive resources and/or time (Feldon, 2007; Guzdial, 2015).

While some evidence suggests that novices cannot spontaneously transfer their knowledge to new domains, some people have had success. For example, Bassok (1990) explored methods of highlighting similarities between algebra and physics problems to promote spontaneous transfer between those fields. Stieff and Uttal (2015) have had success training students in spatial reasoning to improve their performance in STEM fields. In the computing domain, Klahr and Carver (1988) found that teaching elementary school children to debug LOGO programs over several months helped them to complete a number of tasks in various domains, including allocating resources, following a route on a map, and correcting instructions. Computational thinking work has found some evidence that teaching students computing concepts can improve their performance in other domains, like science (Basawapatna *et al.*, 2011; Koh *et al.*, 2010) and math (Schanzer *et al.*, 2015; Schanzer, Fisler, & Krishnamurthi, 2018).

There are many more studies exploring transfer across disciplines, but the research is not conclusive and the arguments are too complex to succinctly describe in a section meant to introduce the transfer literature. In this section, we highlighted some of the foundational work from cognitive science for improving transfer with some examples of how it is applied to CEd.

Much more work is needed, though, to explore various methods of promoting transfer and the conditions in which they are most successful.

7 Cognitive Load

Within an educational setting, we wish to focus the student's attention on the material to be learned. Learning occurs when the information from sensory input is attended to, processed within working memory, and then committed to long term memory in the form of schemata or plans. One interesting theoretical framework that aids the design of effective instructional material is called *cognitive load*. This framework draws together many aspects of cognition, we focus in some detail on it and its practical implications for educators. See also Chapter 3.1 for a discussion of cognitive load in programming.

7.1 What is cognitive load?

Cognitive load is the amount of work imposed on a learner's working memory while learning a specific task (van Gog & Paas, 2012). Cognitive Load Theory (CLT) is based on the architecture of the human brain, with the brain having both working and long-term memory. Working memory is limited in both capacity and duration (Cowan, 2010; Peterson & Peterson, 1959). Learning occurs when new information is related to knowledge already stored and organised in long-term memory, yielding a new more elaborate and extensive knowledge base (Sweller, 1988). The central problem identified by CLT is that learning is impaired when the total amount of processing requirements exceeds the limited capacity of working memory (Plass *et al.*, 2010). Learning is reliant upon the degree of complexity of the new information to be processed, and the way in which that information is presented. Using CLT, instructors attempt to design instructional material in a manner that will not overload working memory, so as to maximise learning.

As originally defined, CLT was the sum of three individual components: 1) *extraneous* load, the effort used to process information that is not fundamental to the concept or process that is being learned; 2) *intrinsic* load, the innate difficulty of the material being learned combined with the learner's characteristics (Leppink *et al.*, 2013); and 3) *germane* load, the effort used to learn the actual material or concept. Recently however, several researchers have questioned whether germane load is a separate component or is actually a piece of the intrinsic load (Kalyuga, 2011). The overall goal behind CLT is to reduce extraneous and intrinsic load, so as to not overload the cognitive process of working memory and impair learning.

Extraneous cognitive load is any information presented to the learner that is not strictly necessary for learning the material or concept. Having redundant information in both text and presented auditorily would be extraneous cognitive load. Extraneous load becomes a problem only when intrinsic load is high (Sweller, 1994). If there is minimal demand on cognitive resources there is ample working memory to handle any extraneous load and learning can still

occur. Worked examples, for instance, contain extraneous information, but if presented correctly, they can aid overall learning.

A topic is considered to have a high intrinsic load if the material being learned is interconnected; that is, learning requires processing several elements simultaneously to understand their relations and interactions (Sweller & Chandler, 1994). In CEd, simply learning the reserved words of a programming language represents a very low interactivity (each is a separate memorisable token), as opposed to understanding all of the elements in a program, where all of the tokens may potentially interact with each other (high interactivity).

Because intrinsic load is essential for comprehending the material and constructing knowledge structures, the instructional material must provide all the components necessary to accommodate this load without exceeding the limits of working memory. Intrinsic load can also vary with the domain expertise and previous knowledge of the learner (Sweller *et al.*, 2011), in that learners with a higher level of previous knowledge may chunk the material differently than novices (Bransford, 2000), allowing them to hold more information in working memory. Thus, the intrinsic load can change based on the learner. An element or a chunk of information for a learner and specific task is determined by the organised knowledge structures or *schemata* in that learner's long-term memory. With the development of expertise, the size of a person's chunks increases, and many interacting elements for a novice become encapsulated into a single element for an expert. The magnitude of the intrinsic cognitive load experienced by a learner is determined by the degree of interactivity of the essential elements relative to the level of learner expertise within the domain (Kalyuga, 2011). This helps to begin to explain the difference between novice and expert programmers.

Sweller now defines germane load in terms of the working memory resources devoted to dealing with intrinsic cognitive load (Sweller, 2018). If, in the material to be learned, the interacting elements are intrinsic to the information being processed, then an intrinsic load is imposed. If multiple elements interact because of an instructional design choice, then extraneous cognitive load is imposed. Germane load is defined in terms of the resources of working memory that are devoted to dealing with the intrinsic load. The more resources that are devoted to intrinsic load, the fewer resources are available for extraneous factors, and higher germane load results. Using this definition, germane cognitive load is dependent on the relationship between the intrinsic and extraneous loads, and is not an independent source of load.

7.2 Empirical Evidence

Research findings supporting CLT principles come from four distinct types of measures:

(1) indirect measures of cognitive load through task performance or accuracy (Ayres, 2001; Cooper & Sweller, 1987) or the time needed for task performance (Chandler & Sweller, 1991, 1992),

(2) dual-task performance measures (Brünken *et al.*, 2004, 2002),

(3) direct physiological measures such as fMRI (Whelan, 2007) or electroencephalographic (EEG) (Antonenko *et al.*, 2010) or eye-tracking variables (van Gog & Scheiter, 2010), and
(4) subjective measurement rating scales (Leppink *et al.*, 2014, 2013; Paas, 1992).

For a thorough treatise on measuring cognitive load, see (Zheng, 2018).

The implications of CLT have led to several different known "effects" that a learner can experience during the learning process. Empirical studies have found that by altering the instructional design materials, learning can be facilitated or impeded. These changes in learning are known as effects. All known effects can be explained using the CLT framework. For a full list and explanation of the effects attributed to cognitive load, see Sweller *et al.* (2011).

Found effects that have implications in CEd include:

- worked example (problem completion) effect - students produce fewer errors and take less time to completion when they study expert solutions, or worked-out solutions, than when they solve the same problems from scratch with no sample solution to follow (Cooper & Sweller, 1987). The same is true for partially solved problems (problem completion) (Paas & van Merriënboer, 1994).
- split-attention effect - information should be presented in a way which is as integrated as possible, both spatially and temporally (Sweller *et al.*, 1990). Examples should imbed explanations rather than separate them into their own text.
- modality effect - this occurs when multiple sources of information that cannot be understood in isolation are needed to be processed simultaneously, usually presented using spoken rather than written text (Tindall-Ford *et al.*, 1997).
- redundancy effect - when unnecessary, additional information is presented to the learner (Chandler & Sweller, 1991). This can occur when duplicate information is presented in both text and verbally.
- expertise reversal effect - occurs when learning material gets in the way of learning for those with prior knowledge (Kalyuga *et al.*, 2003).
- guidance fading effect - states that novices should be provided with many worked examples, followed by completion problems and then full problems (Renkl & Atkinson, 2003; Renkl *et al.*, 2004).
- self-explanation effect - requiring the learner to self-explain new procedures or concepts to enhance learning (Chi *et al.*, 1989; Renkl, 1997).
- element interactivity effect - if intrinsic cognitive load is low, other cognitive load effects cannot be observed; however, if elements are highly interactive, variations in element interactivity can have profound effects on other load effects dependent on extraneous load (Sweller & Chandler, 1994; Tindall-Ford *et al.*, 1997).

7.3 Cognitive Load in Computing

Incorporating CLT into computing education research occurs in the following forms: (1) use of worked examples; (2) use of subgoals; (3) Parsons problems (a specific question format); and (4) attempts to measure cognitive load. The first three of these are all intended to reduce load.

7.3.1 Worked Examples

The first research involving cognitive load in computing was the use of worked examples (Pirolli & Recker, 1994; Recker & Pirolli, 1995) for LISP. In a series of studies designed for implementing and testing a cognitive tutor, they used a form of worked examples to test student knowledge and learning capabilities. The result of the studies showed that students who studied the worked examples, especially with self-explanation skills, performed better than those who simply solved problems or had poor self-explanation skills.

The first extensive study on using a worked example completion format was conducted by van Merriënboer, using introductory computer programming problems (van Merriënboer & Krammer, 1990). A follow-up study (van Merriënboer & De Croock, 1992) examined students who generated programs from scratch compared to those who completed partially constructed programs. For students in the completion group, the presentation of new information and programming practice were linked to incomplete programs, and learners were only required to complete the partial solutions. In the generation group, both model programs and generation assignments were presented, with the model programs serving as the worked example, however, the students were not required to study the model program before beginning generation. The completion group had better post performance.

Casperson and Bennedsen (2007) present a case for using worked examples in computing but provide no empirical evidence validating their use. Skudder and Luxton-Reilly (2014) present sample worked examples that might be used in an introductory programming class with reasoning on why it might be beneficial, but no empirical evidence on use within an actual class. Gray *et al.* (2007) present a detailed set of suggestions for implementing faded worked examples (similar to program completion) for an introductory programming course in C++. They decompose the task of programming into components whose cognitive load can be adequately handled by the students. The decomposition is based on the abstract algorithmic dimensions and the associated concrete programming constructs. The authors provide fully worked examples for all design construct and implementation-construct pairs. However, once again, no empirical evidence exists on using the examples with students in either a laboratory or classroom based study.

Finally, Garner (2002) designed and implemented a Code Restructuring Tool (CORT) to implement code completion tasks based on cognitive load theory and worked examples. The tool was designed to teach the Visual Basic programming language. A quasi-experimental study showed that students receiving the intervention spent less time solving problems, but had no learning gains over the control group.

7.3.2 Subgoals

Another way CEd researchers have attempted to lower the cognitive load of learners is through the use of subgoal labels (see Section 6.2.1 in this chapter). The first known work using

subgoals in computing is (Margulieux *et al.*, 2012). This work used subgoal labels to teach learners how to develop a mobile application using MIT's Android AppInventor. Participants were given a video using subgoal labels as callouts to provide structure to the solution process. The subgoal group attempted and completed successfully more subgoal steps of the assessment tasks, in addition to completing the tasks quicker than the control group. The subgoal group also successfully completed more tasks on a retention task tested one week later. In a second study involving a think-aloud protocol while completing the app-building task the subgoal group outperformed the control group. In addition, Margulieux *et al.* found that the student vocabulary included the subgoals.

Morrison *et al.* (2015) expanded the use of subgoals to learning with a text programming language using loops. The findings indicated that students who learned using subgoals performed better than those who learned without subgoals.

7.3.3 Parsons Problems

One interesting approach to the teaching and assessment of programming is Parsons problems (Parsons & Haden, 2006), in which correct code is broken into code fragments that have to be put in the correct order with the correct indentation. There are several variants, such as including unnecessary code as distractors (Denny *et al.*, 2008). (See Chapter 3.4 for additional information on Parsons Problems.)

Work in this area has found that Parsons problems scores correlate significantly with code writing scores. Parsons problems are simpler than writing code, in particular because students cannot get syntax errors, which lowers cognitive load. Students are more able to focus on issues like meaning and sequencing within problem solving. This means that Parsons problems might be a more efficient way to engage with programming than traditional code writing tasks (Morrison *et al.*, 2016).

7.4 Measuring Cognitive Load

Initial work in this area within CEd was done by Mason *et al.* (2012), and Mason and Cooper (2012). In this work, Mason and colleagues surveyed students in introductory programming courses offered by Australian universities and asked about mental effort. Participants were asked to rate their own levels of mental effort on each of the three components of cognitive load using a 9 point Likert scale. They were also asked to estimate the levels of mental effort on each component experienced by an average student in their introductory programming course and that experienced by a student in the “bottom 10% of performance” in their course; however, the survey questions were never published or validated.

In Morrison *et al.* (2014), the authors adapted an existing cognitive load measurement (Leppink *et al.*, 2013) to computing. They attempted to measure cognitive load components (intrinsic, extraneous, and germane) of learners in a computing environment. The instrument had good internal reliability, but has yet to be replicated.

8 Discussion

This chapter has briefly summarised a range of important topics from cognitive science, noting links with the CEdR literature and speculating about further avenues for research where possible. Apart from the inevitable conclusion that more work is needed, we will return to the concept of levels of analysis (Section 2), and recommend a general framework for describing CEdR:

Pedagogical: This level describes the aims, methods, subject matter and theory of CEd in the abstract. Questions that fall into this level include: What are we trying to achieve? What is known about how individuals, groups and communities construct knowledge? What is the current curriculum and the intended learning outcomes, and what pedagogical methods will best deliver them? What target knowledge, skills and mental models do learners need to be successful? What social and cultural factors affect learning outcomes? What educational concepts and frameworks usefully apply? Research relevant to this level will be being conducted in many disciplines and contexts, but may include specific disciplinary (CEd) topics.

Functional: This level describes the application and interaction of pedagogical and cognitive factors in practical CEd contexts, i.e. as they occur in real classrooms, institutions or other learning environments. Are we achieving our goals? How are learners constructing and sharing their knowledge? Are they making progress with the intended curriculum and outcomes? Are they developing sound mental models? Are techniques for promoting transfer and reducing cognitive load working? What range of variation exists within a given group / population, why, and how can we influence it? How does the particular subject matter and tools of our field impact on teaching and learning? How do institutional context and policy settings affect outcomes? Research relevant to this level is specific to CEd. It is guided, shaped and constrained by both the pedagogical cognitive levels.

Cognitive: This level describes phenomena relating to the cognition and behaviour of individuals. How do individuals represent and acquire knowledge and skills, and how do they develop correct (and incorrect) mental models? What constraints arise from the nature of our learning and memory systems? How do attitudes, motivation and specific behaviours affect learning outcomes? What factors promote transfer and reduce cognitive load? What is the range of variation, and how (and how much) can individuals change or improve? Research relevant to this level will be being conducted generally in the cognitive sciences, but may include specific topics suggested by work at the functional level.

The functional level is the main and characteristic level of our field, but it is informed and constrained by both higher level pedagogical and lower level cognitive factors. When exploring a particular research topic within CEd, we suggest that all levels should be considered. Relevant factors, insights or mechanisms might be found at any level.

Acronyms

CEd	Computing Education
CEdR	Computing Education Research
CLT	Cognitive Load Theory
LTM	Long-Term Memory
SM	Sensory Memory
STM	Short-Term Memory

REFERENCES

ACM/IEEE-CS Joint Task Force on Computing Curricula (2013). *Computer Science Curricula 2013*, USA: ACM Press and IEEE Computer Society Press.

Almstrum, V. L., Hazzan, O., Guzdial, M. & Petre, M. (2005). Challenges to computer science education research. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05)* (pp. 191–192). New York, NY: ACM.

Anderson, J. R. (1976). *Language, memory, and thought,* Hillsdale, NJ: Lawrence Erlbaum.

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review,* **89**, 369–406.

Anderson, J. R., Reder, L. M., & Simon, H. A. (1996). Situated learning and education. *Educational Researcher*, **25**(4), 5–11.

Antonenko, P., Paas, F., Grabner, R., Van Gog, T. (2010). Using electroencephalography to measure cognitive load. *Educational Psychology Review,* **22**, 425–438.

Atkinson, R. C. & Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. In K. W. Spence & J. T. Spence, eds., *Psychology of learning and motivation*, 2. New York, NY: Academic Press, pp. 89–195.

Ayres, P.L. (2001). Systematic mathematical errors and cognitive load. *Contemporary Educational Psychology,* **26**, 227–248.

Baddeley, A. D. (1966). The influence of acoustic and semantic similarity on long-term memory for word sequences. *The Quarterly journal of experimental psychology,* **18**(4), 302–309.

Baddeley, A. D. (1999). *Essentials of Human Memory*, Hove, UK: Psychology Press.

Baddeley, A. D., Eysenck, M. W. & Anderson, M. C. (2015). *Memory,* 2nd ed, London and New York: Psychology Press, Taylor and Francis.

Baddeley, A. D. & Hitch, G. (1974). Working memory. In G. H. Bower, ed., *The Psychology of Learning and Motivation, 2*. New York, NY: Academic Press, pp. 47–89.

Baldwin, L. P., & Macredie, R. D. (1999). Beginners and programming: insights from second language learning and teaching. *Education and Information Technologies,* **4**(2), 167–179.

Bandura, A. (1993). Perceived self-efficacy in cognitive development and functioning. *Educational Psychologist,* **28**(2), 117-148.

Banerjee, A. V., Bhattacharjee, S., Chattopadhyay, R., & Alejandro, J. G. (2017). The Untapped Math Skills of Working Children in India: Evidence, Possible Explanations, and Implications. Downloaded 11 February 2018 from: https://www.alejandroganimian.com/s/Banerjee-et-al-2017-2017-08-17.pdf

Barnett, S. M., & Ceci, S. J. (2002). When and where do we apply what we learn?: A taxonomy for far transfer. *Psychological Bulletin*, **128**(4), 612–637.

Barsalou, L. W. (1983). Ad hoc categories. *Memory & Cognition*, **11**(3), 211-227.

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C. & Marshall, K. S. (2011, March). Recognizing computational thinking patterns. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 245-250). New York, NY: ACM.

Bassok, M. (1990). Transfer of domain-specific problem-solving procedures. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **16**(3), 522–533.

Bassok, M., & Holyoak, K. J. (1989). Interdomain transfer between isomorphic topics in algebra and physics. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **15**(1), 153–166.

Bransford, J. D., Brown, A., & Cocking, R. (1999). *How People Learn: Mind, Brain, Experience, and School.* Washington, DC: National Research Council.

Bereiter, C. (1985). Toward a solution of the learning paradox. *Review of Educational Research,* **55**, 201–226.

Bloom, B., Englehart, M.D., Furst, E.J., Hill, W.H. & Krathwohl, D. (1956). *Taxonomy of Educational Objectives: Handbook I: Cognitive Domain,* NY: Longmans.

Bonar, J. & Soloway, E. (1989) Preprogramming knowledge: a major source of misconceptions in novice programmers. In E. Soloway & J. C. Spohrer, eds., *Studying the Novice Programmer.* Hillsdale NJ: Lawrence Erlbaum, pp. 324–353.

Bouton, M. E. (2016). *Learning and Behavior: A Contemporary Synthesis,* 2nd edn, Sunderland, MA: Sinauer Associates.

Brachman, R. J. & Levesque, H. J., eds. (1985). *Readings in Knowledge Representation,* San Francisco, CA: Morgan Kaufmann Publishers Inc.

Bransford, J. (2000). *How People Learn: Brain, Mind, Experience, and School,* Washington, DC: National Academies Press.

Brown, A. L. (1992). Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, ***2***(2), 141-178.

Brünken, R., Plass, J.L., Leutner, D. (2004). Assessment of cognitive load in multimedia learning with dual-task methodology: Auditory load and modality effects. *Instructional Science,* **32**, 115–132.

Brünken, R., Steinbacher, S., Plass, J.L., Leutner, D. (2002). Assessment of cognitive load in multimedia learning using dual-task methodology. *Experimental Psychology,* **49**, 109–119.

Busjahn, T., Schulte, C., Sharif, B., Begel, A., Hansen, M., Bednarik, R., Orlov, P., Ihantola, P., Shchekotova, G. & Antropova, M. (2014, July). Eye tracking in computing education. In Proceedings of the Tenth Annual Conference on International Computing Education Research (pp. 3-10). New York, NY: ACM.

Cao, L. (2010). In-depth behavior understanding and use: the behavior informatics approach. *Information Sciences,* **180**(17), 3067–3085.

Cao, L., & Philip, S. Y., eds. (2012). *Behavior Computing: Modeling, Analysis, Mining and Decision,* London: Springer Verlag.

Carraher, T. N., Carraher, D. W. & Schliemann, A. D. (1985). Mathematics in the streets and in schools. *British journal of developmental psychology,* **3**(1), 21–29.

Caspersen, M. E., & Bennedsen, J. (2007). Instructional design of a programming course: a learning theoretic approach. In *Proceedings of the Third International Workshop on Computing Education Research* (pp. 111-122). New York, NY: ACM.

Chaiklin, S. (2003). The zone of proximal development in Vygotsky's analysis of learning and instruction. In A. Kozulin, B. Gindis, V. Ageyev & S Miller, eds., *Vygotsky's Educational Theory in Cultural Context, 1.* Cambridge, UK: Cambridge University Press, pp. 39–64.

Chandler, P., Sweller, J. (1992). The split-attention effect as a factor in the design of instruction. *British Journal of Educational Psychology,* **62**, 233–246.

Chandler, P., & Sweller, J. (1991). Cognitive load theory and the format of instruction. *Cognition and instruction,* **8**, 293–332.

Chi, M. T., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science,* **13**(2), 145–182.

Chi, M. T., Feltovich, P. J., & Glaser, R. (1981). Categorization and representation of physics problems by experts and novices. *Cognitive Science*, **5**(2), 121-152.

Chi, M.T.H. & Ohlsson, S. (2005). Complex Declarative Learning. In K.J. Holyoak & R.G. Morrison, eds., *Cambridge Handbook of Thinking and Reasoning*. Cambridge, UK: Cambridge University Press, pp. 371–399.

Cianciolo, A.T. & Sternberg R.J. (2004). *Intelligence: A Brief History,* Oxford, UK: Blackwell Publishing.

Cooper, G., & Sweller, J. (1987). Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology,* **79**, 347–362.

Cooper, J. O. (1982). Applied behavior analysis in education. *Theory Into Practice,* **21**(2), 114-118.

Cooper, P. A. (1993). Paradigm Shifts in Designed Instruction: From Behaviorism to Cognitivism to Constructivism. *Educational technology,* **33**(5), 12-19.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, **24**, 87–185.

Cowan, N. (2010). The magical mystery four: How is working memory capacity limited, and why? *Current Directions in Psychological Science,* **19**, 51–57.

Crystal, D. (2010) *The Cambridge Encyclopedia of Language,* 3rd edn, Cambridge, UK: Cambridge University Press.

Day, S. B., & Goldstone, R. L. (2012). The import of knowledge export: Connecting findings and theories of transfer of learning. *Educational Psychologist*, **47**(3), 153-176.

Deco, G. & Rolls, E.T. (2005). Attention, short-term memory, and action selection: A unifying theory. *Progress in Neurobiology*, **76**, 236–256.

Denny, P., Luxton-Reilly, A., Simon, B. (2008). Evaluating a new exam question: Parsons problems. In *Proceedings of the Fourth International Workshop on Computing Education Research* (pp. 113–124). New York, NY: ACM.

de Winstanley, P. A., Bjork, E. L., & Bjork, R. A. (1996). Generation effects and the lack thereof: The role of transfer-appropriate processing. *Memory,* **4**, 31–48.

Donovan, J. J., & Radosevich, D. J. (1999). A meta-analytic review of the distribution of practice effect: Now you see it, now you don't. *Journal of Applied Psychology,* **84**(5), 795–805.

Dunlosky, J., Rawson, K. A., Marsh, E. J., Nathan, M. J., & Willingham, D. T. (2013). Improving students' learning with effective learning techniques: Promising directions from cognitive and educational psychology. *Psychological Science in the Public Interest*, ***14***(1), 4–58.

Eiriksdottir, E., & Catrambone, R. (2011). Procedural instructions, principles, and examples: how to structure instructions for procedural tasks to enhance performance, learning, and transfer. *Human Factors*, **53**(6), 749–770.

Enderton, H. B. (2001). *A mathematical introduction to logic,* 2nd edn, San Diego, CA: Academic Press.

Engel, A. K., Fries, P. & Singer, W. (2001). Dynamic predictions: oscillations and synchrony in top–down processing. *Nature Reviews Neuroscience*, **2**(10), 704–716.

Eriksen, C. W. & James, J. D. S. (1986). Visual attention within and around the field of focal attention: A zoom lens model. *Perception & Psychophysics*, **40**(4), 225–240.

Ertmer, P. A., & Newby, T. J. (1993). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance improvement quarterly,* **6**(4), 50–72.

Feldman, D.H. (2004). Piaget's stages: the unfinished symphony of cognitive development. *New Ideas in Psychology,* **22**, 175–231.

Feldon, D. F. (2007). The implications of research on expertise for curriculum and pedagogy. *Educational Psychology Review*, **19**(2), 91-110.

Felleisen, M., Findler, R. B., Flatt, M., & Krishnamurthi, S. (2001). *How to Design Programs: An Introduction to Programming and Computing,* MIT Press: Cambridge, MA.

Ferguson, R. W. (1994). MAGI: Analogy-based encoding using regularity and symmetry. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society* (pp. 283-288). London, UK: Psychology Press, Cognitive Science Society.

Ferman S. & Karni A. (2012). Procedural and Declarative Memory in the Acquisition of Morphological Knowledge: A Model for Second Language Acquisition in Adults. In M. Leikin, M. Schwartz, Y. Tobin, eds., *Current Issues in Bilingualism. Literacy Studies (Perspectives from Cognitive Neurosciences, Linguistics, Psychology and Education),* Vol 5. Dordrecht: Springer, pp. 201–216.

Firestone, C. & Scholl, B. J. (2016). Cognition does not affect perception: Evaluating the evidence for "top-down" effects. *Behavioral and Brain Sciences*, **20**, 1–77.

Fischer, K. W. (1980). A theory of cognitive development: The control and construction of hierarchies of skills. *Psychological review,* **87**(6), 477 - 531.

Flavell, J. H. (1992). Cognitive development: Past, present, and future. *Developmental psychology,* **28**(6), 998–1005.

Flavell, J. H. (1996). Piaget's legacy. *Psychological Science,* **7**(4), 200–203.

Foer, J. (2012). *Moonwalking with Einstein: The Art and Science of Remembering Everything.* London: Penguin.

Fougnie, D. (2008). The Relationship between Attention and Working Memory. In N. B. Johansen, ed., *New research on short-term memory*. Hauppauge, NY: Nova Science Publishers, pp. 1–45

Gardner, H. (1985). *The Mind's New Science: A History of the Cognitive Revolution.* New York: Basic Books.

Gardner, H. (1993). *Frames of Mind: The Theory of Multiple Intelligences.* New York: Basic Books.

Garner, S. (2002). *Reducing the Cognitive Load on Novice Programmers,* ERIC ED477013 (7 pages).

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, **7**(2), 155-170.

Gentner, D. (2002). Mental Models, Psychology of. In N. Smelser & P. B. Bates, eds., *International Encyclopedia of the Social and Behavioral Sciences.* Amsterdam: Elsevier Science, pp. 9683–9687.

Gentner, D., Loewenstein, J., & Thompson, L. (2003). Learning and transfer: A general role for analogical encoding. *Journal of Educational Psychology*, **95**(2), 393–408.

Gentner, D. & Stevens, A.L., eds. (1983). *Mental Models,* Hillsdale, NJ: Erlbaum.

Gick, M. L., & Holyoak, K. J. (1980). Analogical problem solving. *Cognitive Psychology*, **12**(3), 306-355.

Gick, M. L., & Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology*, **15**(1), 1-38.

Gilbert, C.D. & Sigman, M. (2007). Brain States: Top-Down Influences in Sensory Processing. *Neuron*, **54**(5), pp. 677–696.

Glass, A. L., Holyoak, K. J. & Santa, J. L. (1979). *Cognition,* Reading, MA: Addison-Wesley.

Goldstein, E. B. & Brockmole, J. (2016). *Sensation and Perception,* 10th edn, CA, USA: Wadsworth, Cengage Learning.

Gray, S., St Clair, C., James, R., Mead, J. (2007). Suggestions for graduated exposure to programming concepts using fading worked examples. In *Proceedings of the Third International Workshop on Computing Education Research* (pp. 99–110). New York, NY: ACM.

Gregory, R. (1997). *Eye and Brain: The Psychology of Seeing,* 5th edn, Oxford, UK: Oxford University Press.

Guzdial, M. (2015). Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, **8**(6), 1-165.

Hampton, J. A. (2016). Categories, prototypes and exemplars. In. N Riemer, ed., *The Routledge Handbook of Semantics* (pp. 125-141). New York: Routledge.

Hansen, L., Umeda, Y., & McKinney, M. (2002). Savings in the relearning of second language vocabulary: The effects of time and proficiency. *Language Learning*, **52**(4), 653-678.

Hansen, M. E., Lumsdaine, A., and Goldstone, R. L. (2013). An experiment on the cognitive complexity of code. In *Proceedings of the Thirty-Fifth Annual Conference of the Cognitive Science Society*. London, UK: Psychology Press, Cognitive Science Society.

Hattie, J. A. (2009). *Visible Learning: A Synthesis of 800+ Meta-Analyses on Achievement,* Abingdon: Routledge.

Hattie, J. (2012). *Visible Learning for Teachers: Maximizing Impact on Learning,* Abingdon: Routledge.

Hazzan, O., Lapidot, T., & Ragonis, N. (2014). Problem-solving strategies. In O. Hazzan, T. Lapidot & N. Ragonis, *Guide to Teaching Computer Science.* London: Springer, pp. 75–93.

Hinton, G.E. & Anderson J. A. (1989). *Parallel Models of Associative Memory,* Hillsdale, NJ: Lawrence Erlbaum Associates Inc.

Holyoak, K. J., & Morrison, R. G., eds., (2005). *The Cambridge Handbook of Thinking and Reasoning.* Cambridge, UK: Cambridge University Press.

Jacoby, L. L. (1978). On interpreting the effects of repetition: Solving a problem versus remembering a solution. *Journal of Verbal Learning and Verbal Behavior*, **17**(6), 649–667.

Jonassen, D. H. (2000). Toward a design theory of problem solving. *Educational Technology Research and Development*, **48**(4), 63-85.

Jensen, A.R. (1998). *The g Factor,* Westport, CT: Praeger.

Johnson-Laird, P. N. (1983). *Mental models: Towards a Cognitive Science of Language, Inference, and Consciousness,* Cambridge, MA: Harvard University Press.

Kalyuga, S. (2011). Cognitive load theory: How many types of load does it really need? *Educational Psychology Review,* **23**, 1–19.

Kalyuga, S., Ayres, P., Chandler, P. & Sweller, J. (2003). The expertise reversal effect. *Educational Psychologist,* **38**, 23-31.

Kapp, K. M. (2012). *The Gamification of Learning and Instruction: Game-Based Methods and Strategies for Training and Education,* San Francisco, CA: John Wiley & Sons.

Kátai, Z., Juhász, K. & Adorjáni, A. K. (2008). On the role of senses in education. *Computers & Education*, **51**(4), 1707–1717.

Killian, S. (2016). Hattie Effect Size 2016 Update. Downloaded 03 February 2018 from http://www.evidencebasedteaching.org.au/hattie-effect-size-2016-update/. Reproduced in Lubelfeld, M., Polyak, N., & Caposey, P. J. (2018), *Student Voice: From Invisible to Invaluable* (pp. 3). Lanham, Maryland: Rowman & Littlefield.

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, **41**(2), 75–86.

Klahr, D., & Carver, S. M. (1988). Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology*, **20**(3), 362–404.

Klaus-Dieter, G. (2018). Eye Movements in Programming Education workshops. Downloaded 16 February 2018 from http://www.mi.fu-berlin.de/en/inf/groups/ag-ddi/Gaze_Workshop/index.html

Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010, September). Towards the automatic recognition of computational thinking for adaptive visual language learning. In *Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 59–66). New York, NY: IEEE.

Kulik, J. A., & Kulik, C. L. C. (1988). Timing of feedback and verbal learning. *Review of Educational Research*, **58**(1), 79–97.

Kurtz, K. J., Miao, C. H., & Gentner, D. (2001). Learning by analogical bootstrapping. *The Journal of the Learning Sciences*, **10**(4), 417–446.

Laird, J. E. & Rosenbloom, P. (1996). The evolution of the Soar cognitive architecture. In D. M. Steier & T. M. Mitchell, eds., *Mind matters: A tribute to Allen Newell.* Mahwah, NJ: Lawrence Erlbaum Associates, pp. 1–50.

Lenneberg, E. H. (1967). The biological foundations of language. *Hospital Practice,* **2**(12), 59–67.

Leppink, J., Paas, F., Van der Vleuten, C.P., Van Gog, T. & van Merriënboer, J.J. (2013). Development of an instrument for measuring different types of cognitive load. *Behavior Research Methods,* **45**, 1058–1072.

Leppink, J., Paas, F., van Gog, T., van der Vleuten, C.P. & van Merriënboer, J.J. (2014). Effects of pairs of problems and examples on task performance and different types of cognitive load. *Learning and Instruction,* **30**, 32–42.

Lindsay, P. H., & Norman, D. A. (1977). *Human Information Processing: An Introduction to Psychology,* 2nd edn, London, UK: Academic Press.

Mandelbaum, E. (2017). Associationist Theories of Thought. *The Stanford Encyclopedia of Philosophy (Summer 2017 Edition),* E. N. Zalta, ed., URL = <https://plato.stanford.edu/archives/sum2017/entries/associationist-thought/>.

Margulieux, L. E., Guzdial, M., & Catrambone, R. (2012, September). Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research* (pp. 71-78). New York, NY: ACM.

Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information,* New York: Freeman.

Marr D. & Poggio T. (1976). From Understanding Computation to Understanding Neural Circuitry. *A.I. Memo 357*. Massachusetts Institute of Technology: Cambridge, MA.

Marshall, S. P. (1995). *Schemas in Problem Solving*, Cambridge University Press.

Mason, R., Cooper, G. (2012). Why the bottom 10% just can't do it: mental effort measures and implication for introductory programming courses. In *Proceedings of the Fourteenth Australasian Computing Education Conference,* Volume 123, (pp. 187–196). Sydney, NSW: Australian Computer Society.

Mason, R., Cooper, G., de Raadt, M. (2012). Trends in introductory programming courses in Australian universities: languages, environments and pedagogy. In *Proceedings of the Fourteenth Australasian Computing Education Conference*, Volume 123, (pp. 33–42). Sydney, NSW: Australian Computer Society.

Mathan, S., & Koedinger, K. R. (2003). Recasting the feedback debate: Benefits of tutoring error detection and correction skills. In *Proceedings of the International Conference on Artificial Intelligence in Education* (pp. 13-20). Amsterdam: IOS Press.

Mather, G. (2016). *Foundations of Sensation and Perception,* 3rd end, London & New York: Routledge.

McClamrock, R. (1991). Marr's three levels: A re-evaluation. *Minds and Machines,* **1**(2), 185–196.

McNeill, D. (1970). *The Acquisition of Language: The Study of Developmental Psycholinguistics,* New York & London: Harper and Row.

McSweeney, F. K., & Murphy, E. S., eds (2014). *The Wiley Blackwell Handbook of Operant and Classical Conditioning,* Madden, MA: John Wiley & Sons.

Medin, D. L., Altom, M. W. & Murphy, T. D. (1984). Given versus induced category representations: Use of prototype and exemplar information in classification. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **10**(3), 333–352.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, **63**, 81–97.

Minda, J. P. & Smith, J. D. (2002). Comparing prototype-based and exemplar-based accounts of category learning and attentional allocation. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **28**(2), 275–292.

Morra, S., Gobbo, C., Marini, Z. & Sheese, R. (2007) *Cognitive Development: Neo-Piagetian Perspectives,* New York, NY: Psychology Press.

Morris, M. R., Begel, A., & Wiedermann, B. (2015, October). Understanding the challenges faced by neurodiverse software engineering employees: Towards a more inclusive and productive technical workforce. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility* (pp. 173-184). New York: NY: ACM.

Morrison, B.B., Dorn, B., Guzdial, M. (2014). Measuring Cognitive Load in Introductory CS: Adaptation of an Instrument. In *Proceedings of the Tenth Annual Conference on International Computing Education Research (ICER '14)* (pp. 131–138). New York, NY: ACM.

Morrison, B. B., Margulieux, L. E., & Guzdial, M. (2015, July). Subgoals, context, and worked examples in learning computing problem solving. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (pp. 21-29). New York, NY: ACM.

Morrison, B.B., Margulieux, L.E., Ericson, B., Guzdial, M. (2016). Subgoals Help Students Solve Parsons Problems. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)* (pp. 42–47). New York, NY: ACM.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving,* Englewood Cliffs, NJ: Prentice-Hall.

Newport, E. L. (1990). Maturational constraints on language learning. *Cognitive science,* **14**(1), 11–28.

Norman, D. A., ed. (1970). *Models of Human Memory*, New York & London: Academic Press.

Novick, L. R., & Bassok, M. (2005). Problem solving. In K. J. Holyoak & R. G. Morrison, eds., *Cambridge Handbook of Thinking and Reasoning.* Cambridge, UK: Cambridge University Press, pp. 321–349.

Paas, F.G. (1992). Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach. *Journal of Educational Psychology,* **84**, 429–434.

Paas, F. G. & van Merriënboer J. J. (1994). Variability of worked examples and transfer of geometrical problem-solving skills: A cognitive-load approach. *Journal of Educational Psychology,* **86**(1), 122–133.

Pandža N.B. (2016) Computer Programming as a Second Language. In D. Nicholson, ed., *Advances in Human Factors in Cybersecurity. Advances in Intelligent Systems and Computing, Vol 501* (pp. 439–445). Switzerland: Springer, Cham.

Parsons, D. & Haden, P. (2006). Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52 (ACE '06)* (pp. 157–163). Darlinghurst, Australia: Australian Computer Society.

Pashler, H., ed. (2016). *Attention,* NY: Psychology Press.

Pea, R.D. & Kurland, D.M. (1984). On the Cognitive Prerequisites of Learning Computer Programming. *Technical Report No.18*. New York, NY: Bank Street College of Education.

Peterson, L., Peterson, M.J. (1959). Short-term retention of individual verbal items. *Journal of Experimental Psychology,* **58**, 193–198.

Piaget, J. (1952). *The Origins of Intelligence in Children,* New York: International Universities Press.

Piaget, J. (1964). Part I: Cognitive development in children: Piaget development and learning. *Journal of Research in Science Teaching,* **2**(3), 176 – 186.

Piaget, J. (1971a). The theory of stages in cognitive development. In D.R. Green, M.P. Ford, & G.B. Flamer, eds., *Measurement and Piaget*. NY: McGraw-Hill, pp. 1–11.

Piaget, J. (1971b). Developmental stages and developmental processes. In D.R. Green, M.P. Ford, & G.B. Flamer, eds., *Measurement and Piaget.* NY: McGraw-Hill, pp. 172–188.

Pirolli, P., Recker, M. (1994). Learning strategies and transfer in the domain of programming. *Cognition and Instruction,* **12**, 235–275.

Plass, J.L., Moreno, R., Brünken, R. (2010). *Cognitive Load Theory,* Cambridge: Cambridge University Press.

Pohl, R., ed., (2004). *Cognitive Illusions: A Handbook on Fallacies and Biases in Thinking, Judgement and Memory,* Hove, UK: Psychology Press.

Posner, M. I., Snyder, C. R. & Davidson, B. J. (1980). *Attention and the detection of signals. Journal of Experimental Psychology: General*, **109**(2), 160–174.

Powell, S. R. (2011). Solving word problems using schemas: A review of the literature. *Learning Disabilities Research & Practice*, ***26***(2), 94–108.

Quine, W. V. (1969). Natural kinds. In C. G. Hempel, D. Davidson & N. Rescher, eds., *Essays in Honor of Carl G. Hempel*. Dordrecht: Springer, pp. 5–23.

Radvansky, G. (2016). *Human Memory (Second Edition)*. New York: Routledge.

Recker, M. & Pirolli, P. (1995). Modeling individual differences in students' learning strategies. *The Journal of the Learning Sciences,* **4**, 1–38.

Reed, S. K., Ernst, G. W. & Banerji, R. (1974). The role of analogy in transfer between similar problem states. *Cognitive Psychology*, **6**(3), 436–450.

Renkl, A. (2017). Learning from worked-examples in mathematics: students relate procedures to principles. *ZDM Mathematics Education,* **49**(4), *571-584.*

Renkl, A., Atkinson, R., & Grosse, C. (2004) How fading worked solution steps works – a cognitive load perspective. *Instructional Science,* **32**, 59-82.

Renkl, A. & Atkinson, R. (2003). Structuring the transition from example study to problem solving in cognitive skill acquisition: A cognitive load perspective. *Educational Psychologist.* **38**(1), 15–22.

Renkl, A. (1997). Learning from worked-out examples: A study on individual differences. *Cognitive Science*, **21**, 1–29.

Revlin, R. (2012). *Cognition: Theory and Practice*, New York, NY: Worth Publishers.

Rieber, R.W. & Robinson D.K., eds. (2004). *The Essential Vygotsky,* NY: Kluwer Academic / Plenum Publishers.

Rist, R. S. (1995). Program Structure and Design. *Cognitive Science,* **19***,* 507 – 562.

Robertson, S. I. (2016). *Problem Solving: Perspectives From Cognition and Neuroscience,* New York, NY: Rutledge.

Robins, A. (1996). Consolidation in neural networks and in the sleeping brain. *Connection Science*, **8**(2), 259–276.

Robins A. (2010). Learning Edge Momentum: A New Account of Outcomes in CS1. *Computer Science Education***, 20**, 37–71.

Rosch, E., Mervis, C. B., Gray, W. D., Johnson, D. M. & Boyes-Braem, P. (1976). Basic objects in natural categories. *Cognitive psychology*, **8**(3), 382–439.

Rosenbloom, P. S., Laird, J. E., Newell, A. & McCarl, R. (1991). A preliminary analysis of the Soar architecture as a basis for general intelligence. *Artificial Intelligence,* **47**(1-3), 289–325.

Schacter, D. L., Addis, D. R. & Buckner, R. L. (2007). Remembering the past to imagine the future: the prospective brain. *Nature Reviews Neuroscience*, **8**(9), 657–661.

Schank, R.C. & Abelson, R.P. (1975). Scripts, plans, and knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence (IJCAI'75) - Volume 1* (pp. 151-157). San Francisco, CA: Morgan Kaufmann Publishers.

Schank, R. C. & Abelson, R. P. (2013). *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*, NY: Psychology Press.

Schanzer, E., Fisler, K. & Krishnamurthi, S. (2018). Assessing Bootstrap: Algebra Students on Scaffolded and Unscaffolded Word Problems. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)* (pp. 8-13). New York, NY: ACM.

Schanzer, E., Fisler, K., Krishnamurthi, S., & Felleisen, M. (2015, February). Transferring skills at solving word problems from computing to algebra through bootstrap. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 616-621). New York, NY: ACM.

Schoenfeld, A. H. (1985). *Mathematical Problem Solving,* London, UK: Academic Press.

Scholnick, E. K., Nelson, K., Gelman, S. A. & Miller, P. H., eds. (1999). *Conceptual Development: Piaget's legacy,* New York: Psychology Press.

Shams, L. & Seitz, A. R. (2008). Benefits of multisensory learning. *Trends in Cognitive Sciences*, **12**(11), 411–417.

Singer, S. R., Nielsen, N. R. & Schweingruber, H. A., eds. (2012). *Discipline-Based Education Research: Understanding and Improving Learning in Undergraduate Science and Engineering,* Washington, DC: National Academies Press.

Singleton, D. M. & Ryan, L. (2004). *Language Acquisition: The Age Factor,* Clevendon, UK: Multilingual Matters.

Singley, M. K., & Anderson, J. R. (1989). *The Transfer of Cognitive Skill* (No. 9), Harvard: Harvard University Press.

Skinner B. F. (1938). *The Behavior of Organisms: An Experimental Analysis,* Oxford, England: Appleton-Century.

Skinner B. F. (1953). *Science and Human Behavior,* New York: Macmillan.

Skudder, B., Luxton-Reilly, A. (2014). Worked examples in computer science. In *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148* (pp. 59–64). Sydney, NSW: Australian Computer Society.

Smith, T. J. (2007). The ergonomics of learning: educational design and learning performance. *Ergonomics*, **50**(10), 1530–1546.

Smith, T. A., & Kimball, D. R. (2010). Learning from feedback: Spacing and the delay–retention effect. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **36**(1), 80–95.

Soar (2018). Soar Home. Downloaded on 6 February 2018 from: https://soar.eecs.umich.edu/

Sowa, J. F. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations,* Pacific Grove, CA: Brooks/Cole.

Sternberg, R.J., ed. (1982). *Handbook of Human Intelligence,* New York: Cambridge University Press.

Stickgold, R. (2005). Sleep-dependent memory consolidation. *Nature*, ***437***(7063), 1272–1278.

Stieff, M. & Uttal, D. (2015). How much can spatial training improve STEM achievement?. *Educational Psychology Review*, **27**(4), 607–615.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction (Vol. 1, No. 1)*, Cambridge: MIT Press.

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review*, **22**(2), 123–138.

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction,* **4**, 295–312.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science,* **12**, 257–285.

Sweller, J. (2018). The role of independent measures of load in cognitive load theory. In R. Z. Zheng, ed., *Cognitive load measurement and application.* New York, NY: Routledge, pp. 17–22.

Sweller, J., Ayres, P. & Kalyuga, S. (2011). *Cognitive Load Theory,* New York, NY: Springer.

Sweller, J., Chandler, P. (1994). Why some material is difficult to learn. *Cognition and Instruction,* **12**, 185–233.

Sweller, J., Chandler, P., Tierney, P. & Cooper, M. (1990) Cognitive load as a factor in the structuring of technical material. *Journal of Experimental Psychology: General*, **119**, 176–192.

Szpunar, K. K., McDermott, K. B. & Roediger, H. L. (2007). Expectation of a final cumulative test enhances long-term retention. *Memory & Cognition*, **35**(5), 1007–1013.

Thorndike, E. (1911). *Animal intelligence: Experimental studies,* New York: Macmillan.

Tindall-Ford, S., Chandler, P. & Sweller, J. (1997). When two sensory modes are better than one. *Journal of Experimental Psychology: Applied*, **3**(4), 257–287.

Tobias, S. & Duffy, T. M., eds. (2009). *Constructivist Instruction: Success or Failure?*, New York: Routledge.

Tonegawa, S., Pignatelli, M., Roy, D. S. & Ryan, T. J. (2015). Memory engram storage and retrieval. *Current Opinion in Neurobiology,* **35**, 101–109.

Trafton, J. G. & Reiser, B. J. (1993). Studying examples and solving problems: Contributions to skill acquisition. In *Proceedings of the 15th conference of the Cognitive Science Society* (pp. 1017-1022). London, UK: Psychology Press, Cognitive Science Society.

Treisman, A. M. & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology*, **12**(1), 97–136.

Tulving, E. (1985). *Elements of Episodic Memory*, Oxford, England: Clarendon Press.

Tulving, E. & Thomson, D. M. (1973). Encoding specificity and retrieval processes in episodic memory. *Psychological Review*, **80**(5), 352–373.

Tulving, E. (1974). Cue-dependent forgetting: When we forget something we once knew, it does not necessarily mean that the memory trace has been lost; it may only be inaccessible. *American Scientist*, **62**(1), 74–82.

Tversky, A., & Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science,* **185** (4157), 1124–1131.

Twyman, J. S. (2014). Behaviour analysis in education. In F. K.McSweeney & E. S. Murphy, eds., *The Wiley Blackwell Handbook of Operant and Classical Conditioning.* Madden, MA: John Wiley & Sons, pp. 553 - 558.

Ullman, M. T. (2001). A neurocognitive perspective on language: The declarative/procedural model. *Nature Review Neuroscience,* **2**, 717–726.

Ullman, M. T. (2004). Contribution of memory circuits to language: The declarative/procedural model. *Cognition,* **92**, 231–270.

van Gog, T. & Paas, F. (2012). Cognitive Load Measurement. In N. M. Seel, ed., *Encyclopedia of the Sciences of Learning.* New York, NY: Springer, pp. 599–601.

van Gog, T. & Scheiter, K. (2010). Eye tracking as a tool to study and enhance multimedia learning. *Learning and Instruction,* **20**, 95–99.

van Merriënboer, J.J. & Krammer, H.P. (1990). The “completion strategy” in programming instruction: Theoretical and empirical support. In S. Dijkstra, B.H.A.M. van Hout Wolters, P.C. van der Sijde, eds., *Research on Instruction: Design and Effects* (pp. 45-61). Englewood Cliffs, NJ: Educational Technology Publications.

van Merriënboer, J.J. & De Croock, M.B. (1992). Strategies for computer-based programming instruction: Program completion vs. program generation. *Journal of Educational Computing Research,* **8**, 365–394.

van Merriënboer, J. J., Clark, R. E., & De Croock, M. B. (2002). Blueprints for complex learning: The 4C/ID-model. *Educational Technology Research and Development*, **50**(2), 39–61.

Volder, J. E. (1959). The CORDIC trigonometric computing technique. *IRE Transactions on Electronic Computers*, **3**, 330–334.

Wang, Y., & Chiew, V. (2010). On the cognitive process of human problem solving. *Cognitive Systems Research,* **11**(1), 81–92.

Watson, J.B. (1913). Psychology as the behaviorist views it. *Psychological Review,* **20**, 158–177.

Watson, J. B. (1930). *Behaviorism,* New York: Norton.

Weiser, M., & Shertz, J. (1983). Programming problem representation in novice and expert programmers. *International Journal of Man-Machine Studies*, **19**(4), 391–398.

Wertsch, J. V. (1984). The zone of proximal development: Some conceptual issues. *New Directions for Child and Adolescent Development,* **23**, 7–18.

Wertsh, J. V. & Tulviste, P. (1990). Apprenticeship in thinking: Cognitive development in social context. *Science,* **249**(4969), 684–686.

Whelan, R.R. (2007). Neuroimaging of cognitive load in instructional multimedia. *Educational Research Review,* **2**, 1–12.

Whitney, P. (2001). Schemas, frames, and scripts in cognitive psychology. In N. J. Smelser & P. B. Baltes, eds, *International Encyclopedia of the Social & Behavioral Sciences*. Amsterdam: Elsevier, pp. 13522–13526

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, **49**(3), 33–35.

Wood, D., Bruner, J. S. & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry,* **17**(2), 89–100.

Woollard, J. (2010). *Psychology for the Classroom: Behaviourism,* New York, NY: Routledge.

Zheng, R. Z. (2018). *Cognitive Load Measurement and Application: A Theoretical Framework for Meaningful Research and Practice,* New York: NY: Routledge.