Informational Equivalence but Computational Differences? Herbert Simon on Representations in Scientific Practice

David Waszek

Author's accepted manuscript. Published in Minds and Machines, 2023.
DOI: https://doi.org/10.1007/s11023-023-09630-4

Informational equivalence but computational differences? Herbert Simon on representations in scientific practice

David Waszek

December 31, 2022

Abstract

To explain why, in scientific problem solving, a diagram can be "worth ten thousand words," Jill Larkin and Herbert Simon (1987) relied on a computer model: two representations can be "informationally" equivalent but differ "computationally," just as the same data can be encoded in a computer in multiple ways, more or less suited to different kinds of processing. The roots of this proposal lay in cognitive psychology, more precisely in the "imagery debate" of the 1970s on whether there are image-like mental representations. Simon (1972, 1978) hoped to solve this debate by thoroughly reducing the differences between forms of mental representations (e.g., between images and sentences) to differences in computational efficiency; to carry out this reduction, he borrowed from computer science the concepts of data type and of data structure. I argue that, in the end, his account amounted to nothing more than characterizing representations by the fast operations on them. This analysis then allows me to assess what Simon's approach actually achieves when transported from psychology to the study of scientific representations, as in Larkin and Simon (1987): it allows comparing, not representations in and of themselves, but rather the computational roles they play in particular problem-solving processes—that is, representations together with a particular way of using them.

Introduction

It has become commonplace in contemporary philosophy of science to emphasize that, in scientific practice, "representations matter" (Humphreys 2004: 98)—drawing a well-chosen diagram or introducing a suitable symbolic notation, for instance, can be of considerable help in problem solving. To make sense of this, Herbert Simon suggested thinking of representations on the model of data storage in computers (Simon 1978; Larkin and Simon 1987): just as the same data can be encoded in different ways, making certain ways of processing them more or less efficient, so two representations can be "informationally" equivalent but differ "computationally."

In our days of pervasive computing, such a picture of representational differences seems quite natural and has proven popular, with or without explicit reference to Simon,[1] but it is rarely spelled out and its difficulties are not usually discussed. By reconstructing and contextualizing Simon's research program on representations, this paper aims at clarifying what it does and does not achieve.

As we shall see, the original context of Simon's work on representations was cognitive psychology, more precisely a debate on whether there were image-like mental representations (Section 1); to this debate, Simon brought a very specific methodology, based on the study of human problem solving through computer simulations (Section 2). This twofold context sheds light on Simon's 1978 attempt to clarify what it means to represent something in a certain form (e.g., "as an image"): he needed an account of representational forms that could be operationalized in psychological experiments, but would also be broadly applicable to the (not necessarily mental) representations used in problem solving; to get there, he naturally turned to the ways data are stored in computers.

Simon's idea, which is introduced informally in Section 3, is roughly that (1) two ways of storing data can be called informationally equivalent if they are algorithmically intertranslatable, and that (2) such informationally equivalent ways to store data can differ in their computational properties, that is, in the kinds of processing they facilitate. Further, he thought that on this basis, one could fully characterize representational forms *abstractly* by their computational properties, and that this would permit a full empirical solution to the imagery debate.

To buttress his computational account of representational differences, Simon relied on concepts from computer science, in particular those of *(abstract) data type* and of *data structure*. I present these concepts and argue that in the end, it amounts to nothing more than characterizing representations by the fast operations on them (Section 4), though Simon's formulations and practice sometimes suggest a more substantial account (Section 5).

Simon's abstract, purely operations-based account stands in sharp contrast with more standard understandings of representations, which see them as material entities that are individuated syntactically, and endowed with a semantics, by virtue of their belonging to some kind of representational system (one might think of Goodman's notion of *symbolic system*,[2] or of Haugeland's notion of *representational scheme*,[3] though details do not matter here).

[1] Larkin and Simon (1987) remains one the most cited papers ever published by the journal *Cognitive Science*. For recent examples of references to their analysis in the philosophy of science, see Vorms (2011; 2012) or Kulvicki (2010: 300). Humphreys (2004: 95–100) does not mention Simon explicitly, but his discussion of representations is very much in Simon's spirit: in fact, although Humphreys seems to have learned of it indirectly, the most striking example he uses—a game that an appropriate change of representation shows to be a variant of tic-tac-toe—was initially designed by Simon (1969: 76), who called it "number scrabble" (see also the latest edition: Simon 1996: 131).

[2] Goodman (1968).

[3] Haugeland (1998).

Jettisoning such standard conceptions in favor of Simon's, which means studying representations while ignoring matters of physical realization, can make sense for mental representations—and indeed, this is now standard practice in cognitive psychology. For the external representations used in scientific problem solving, however, such a move is not plausible. In this context, instead of fully reducing representational differences to computational differences, the effect of Simon's computational approach is rather to shift the topic: it only allows comparing representations *relative to a particular way of using them*. This shift becomes clear when examining Larkin and Simon's work on scientific representations in light of the preceding discussion (Section 6). In the end, despite what Simon's slogan might suggest, representational differences in scientific practice cannot be accounted for in a purely computational way.

1 The context (1): cognitive science and the imagery debate

Let us first survey the immediate context of Simon's work on representations. It lies in what we now call "cognitive science," and specifically in the "mental imagery" debate that shook the field in the 1970s (and beyond).

The "cognitive science" label is somewhat anachronistic here. Roughly, from the 1950s onward, there arose a number of new research programs on human behavior that shared the following postulates, linked to their reliance on computers as a model of cognition.[4] First, "what goes on inside the individual" can be studied and described not only at the physiological (and in particular neurophysiological) level, *but also* at a more abstract level, in terms of information processing—just as a computer can be described, not only at a material level, but also at the "informational" level, that is, in terms of programs. (In the philosophy of mind, this first postulate is known as *functionalism*.) Second, this "informational" description is phrased in terms of representational mental states or "mental representations," and of processes operating on these representations. Many authors push the computer model even further, and add a third postulate, namely that the mental representations just discussed are systems of *symbols*—and the processes operating on them are *effective*—in the sense of computability theory. This strong thesis, henceforth called the "symbolic view" of cognition, implies that human cognition can be fully simulated on a computer; it was particularly important for artificial intelligence (AI), a field which played a structuring role in the emergence of cognitive science. Herbert Simon is one of the pioneers of AI, and—together with Allen Newell—authored its

[4] It is common to speak of "cognitivism" (Haugeland 1978) or of a cognitive "paradigm" unifying researchers from various disciplines. The best starting point to explore the field thus delimited is probably Haugeland (1981). From a historical point of view, however, one should keep in mind that the unity of what eventually became "cognitive science" was constructed progressively and did not become fully apparent until the 1970s. In addition, the choice to isolate as essential the postulates that I list here (following usual practice) is tied to a philosophically oriented vision of cognitive science that gives theoretical pride of place to AI.

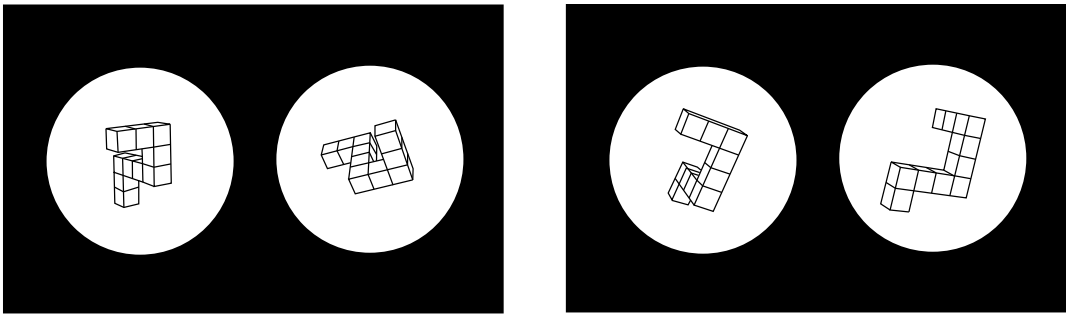

Figure 1: Two pairs of drawings used by Shepard and Metzler (1971): on the left, two figures that can be obtained one from the other by rotation; on the right, two figures that cannot

most famous manifesto.[5]

In this context, in which the existence of "mental representations" is taken for granted, the imagery debate is about the *form* of these representations: are they all language-like or are some of them image-like, and what does the distinction even mean? Here is Simon's formulation of the problem:

> The human brain encodes, modifies, and stores information that is received through its various sense organs, transforms that information by the processes that are called "thinking," and produces motor and verbal outputs of various kinds based on the stored information. So much is noncontroversial [. . .]. What *is* highly controversial is *how* information is stored in the brain—in the usual terminology, how it is "represented"—or even how we can describe representations, and what we mean when we say that information is represented in one way rather than another.[6]

More specifically, the debate was ignited by a number of results coming from experimental psychology, the most famous of which is due to Roger Shepard and Jacqueline Metzler (1971). They displayed pairs of drawings of three-dimensional objects (Fig. 1) to experimental subjects, whom they asked in each case whether the objects presented could be obtained one from the other by rotation. Shepard and Metzler then measured the subjects' response time, and that is where the important result arose: they discovered that the amount of time necessary before getting an affirmative answer was, on average, proportional to the angle of the rotation between the two objects presented—as if the subjects were performing a *mental rotation* at a fixed speed. A long and lively controversy ensued over how to interpret various results that, like this one, seemed to suggest the existence of "mental images," a concept many critics thought was unnecessary, inconsistent and based on a confused

[5] Newell and Simon (1976), reprinted in Haugeland (1981).
[6] Simon (1978: 3).

analogy with "external" images (like pictures and photographs).[7]

Simon's position in this debate is nuanced. As the archetypal advocate of the "symbolic" view of cognition, he is sometimes assumed to be an obvious opponent of mental imagery, but this is a misunderstanding. In fact, the main strand of the imagery debate played out *among* researchers committed to the symbolic view. Roughly, one side—most famously Steven Kosslyn—defended the existence of image-like mental representations that were close to raw perception and could be modeled (symbolically) by arrays of pixels. The other side, spearheaded by Zenon Pylyshyn, thought the experimental data could and should be explained by mental representations that were at a remove from raw perception, and instead were closer to higher-level descriptions based on concepts (though they need not be expressible in sentences of natural language).[8] (Pylyshyn called such representations "abstract"; Simon called them "semantic" or "conceptual.") While sympathetic to Pylyshyn's view, Simon was never fully committed to it, and treated the matter as an open question to be investigated experimentally.[9]

In any case, Simon (1978) addressed a somewhat different question: the deeper and more interesting image–sentence contrast, for him, lay *among* representations that would be considered "abstract" or "semantic" in the standard imagery debate. To make sense of this, one should go back to his earlier AI programs, developed in collaboration with Allen Newell. Newell and Simon's programs were, essentially, problem solvers; their first one was designed to find proofs for logical theorems.[10] Yet even in logical contexts, the data stores of these programs (their "mental representations," so to speak) were not made of sentences of some natural or logical language, but rather of something the authors eventually called list structures—essentially, graphs.[11] List structures could be used to encode sentences, but they were much more general. As a simple example, think of a living-room scene. You could describe it by a series of sentences ("There is a glass vase on the table," etc.); instead, you could represent the scene as a list structure whose nodes are objects (e.g., "glass vase," "table") and whose links are relations between these objects (e.g., a link encoding the spatial relation "on" between the vase and the table). From the point of view of the Kosslyn–Pylyshyn debate, the distinction is irrelevant: both representations are "abstract"—in our example, both refer to something conceptualized as a "glass vase," not to a detailed array of pixels. (Indeed, Pylyshyn explicitly argued

[7] For an introduction to this debate, see Thomas (2014: section 4.4).

[8] See, e.g., Pylyshyn (1973) and Kosslyn (1980). Thomas (2014) provides a full bibliography of the debate.

[9] In Simon (1978: 16), for example, he explicitly allowed that a "specifically visual system" may be needed to account for some of the mind's geometrical capabilities—an open-minded position he maintained consistently (see, e.g., Simon 1989: 384).

[10] This is their "Logic Theory Machine," developed in late 1955 and published in 1956 (Newell and Simon 1956; 1963). MacKenzie (2001: chap. 3) gives a readable and well-contextualized account of this work; see also Dick (2015).

[11] See, e.g., Newell and Simon (1972).

that, for his purposes, Simon's list structures could be assimilated to sentences.)[12] However, Simon's insight, fully articulated as early as 1972, is that a list structure that is "abstract" or "semantic," like our living-room graph, can nevertheless *behave like an image*: one can have "pictorial-semantic" representations just as well as "linguistic-semantic" ones.[13] For him, the abstractness of a representation was independent of whether it is image-like or sentence-like, making his contribution orthogonal to the main line of the imagery debate.

The contrast between image-like and sentence-like list structures mattered greatly to Simon also beyond the imagery debate. His preference for non-sentential structures positioned his work with respect to competing AI research programs; as he put it in his autobiography:

> An influential coterie of contemporary artificial intelligence researchers, including Nils Nilsson, John McCarthy, and others, believe that formal logic provides the appropriate language for A.I. programs, and that problem solving is a process of proving theorems. They are horribly wrong on both counts [. . .].[14]

Accordingly, defending the power of diagrams eventually amounted, for him, to defending his entire approach to AI.[15]

Explicating the image–sentence contrast, then, was doubly important to Simon. To understand the account he offered, a detour through his earlier work is in order.

2 The context (2): Simon, heuristics, and problem solving

To the imagery question, Simon brought an outlook and methodology that were quite peculiar, even among AI researchers.[16] Let us quickly survey his earlier work.

Simon's career started not in AI, but in political science, and more precisely in the theory of organizations. Even today, he remains quoted above all for his concept of "bounded rationality," which was designed to criticize the idealized models of a fully rational agent with complete information that used to dominate economics. As he put it in 1947:

> It is impossible for the behavior of a single, isolated individual to reach any high degree of rationality. The number of alternatives he must ex-

[12] Pylyshyn (1973: 14).

[13] See Simon (1972: 197–200).

[14] Simon (1991: 192, note).

[15] See Simon (1989: 383–384).

[16] For an overview of Simon's broad and multidisciplinary career, see Crowther-Heyck (2005). On his AI work, there are numerous sources, including his autobiography (Simon 1991), Pamela McCorduck's collection of interviews (2004), and Crevier (1993). MacKenzie (2001: chap. 3) gives an excellent introduction to Simon's work on automated proof in the 1950s and 1960s. See also Mirowski (2002: 452–479, 529–533) for a stimulating reevaluation.

> plore is so great, the information he would need to evaluate them so vast that even an approximation to objective rationality is hard to conceive.[17]

How can one orient oneself in this vast space of possibilities? In his early work, Simon concluded that organizations have to constrain the agents' environment by limiting their possibilities and by only providing them with the most relevant information; later, he concluded more generally that human rationality involves finding strategies to orient oneself in a space of possibilities that would otherwise be too vast—strategies that, while not optimal, can still be efficient.[18]

When Simon (in collaboration with Allen Newell) turned to AI—that is, for starters, to the development of chess-playing and theorem-proving programs—he employed strikingly similar turns of phrase:

> In a fundamental sense, proving theorems and playing chess involve the same problem: reasoning with heuristics that select fruitful paths of exploration in a space of possibilities that grows exponentially.[19]

While the use of the word "heuristic" is new here (it seemingly comes from the mathematician George Pólya, whose lectures Allen Newell had attended),[20] the idea remains similar: what matters is finding solutions that are accessible to us despite the extreme limitation of our capabilities. Crucially, this led Simon to the idea of *simulating* human heuristics and (more broadly) human problem-solving processes. In his view, AI was not just about building machines capable of accomplishing complex tasks, but also about doing this while respecting the kinds of limitations that constrain human problem solving—and thus about imitating human strategies. Accordingly, his AI programs were at least as much about shedding light on human cognition as about artificially reproducing human cognitive achievements.

Now, as Donald MacKenzie (2001: chap. 3) compellingly shows, this conception of AI was not widely shared among Simon's colleagues, who thought that the best way *for a computer* to solve a problem might be very different from the best way *for a human* to do it. As Newell later put it, in an interview from the early 1990s:

> In Herbert [Simon]'s and my stuff [. . .] the concern for artificial intelligence [is always] right there with the concern for cognitive psychology. Large parts of the rest of the AI field believe that is exactly the wrong way to look at it: you ought to know whether you're being a psychologist or an engineer. Herb and I have always taken the view that maximal confusion between those is the way to make progress.[21]

[17] Simon (1947: 79).

[18] On the genesis of the concept of "bounded rationality," see Crowther-Heyck (2005) and Mirowski (2002: 456 *sq*). Dick (2015) explores how it was inflected by Simon's practice of computer simulations, discussed below.

[19] Newell, Shaw and Simon (1963: 50).

[20] Simon (1991: 199).

[21] Crevier (1993: 258).

This peculiarity of Simon's approach brings us back to the problem of representations. Indeed, while his and Newell's first success in AI is their well-known "Logic Theory Machine,"[22] it turns out that their initial project was to construct a theorem-prover for *geometry* rather than logic, and that this had already led them to think about the heuristic role of geometrical diagrams. (They switched to logic fairly late in the process, because, as Simon put it, "[f]acing the question of how to do the diagrams, we saw that we still had to deal with some basic problems of perception.")[23] A few months after our authors left it aside, the question of geometrical theorem proving popped up again during the so-called "Dartmouth conference," a summer meeting that brought together many pioneers of AI, including Newell and Simon but also Marvin Minsky and John McCarthy.[24] Minsky suggested a diagram-based heuristic method to find proofs of geometrical theorems;[25] his idea was taken up by Herbert Gelernter (1963), who designed a fairly efficient program that relied on diagrams (implemented as arrays of pixels).

Thus, already in the 1950s, Simon was mulling over the idea that appropriate representations could provide efficient heuristics and thus offer a computational advantage in problem solving. The design of efficient representations would remain one of his most consistent concerns throughout his AI career—a concern he liked to contrast with competing logic-based approaches to AI, which, rather than designing tailor-made representations that allowed for efficient heuristics, systematically relied on sentences of logical languages and on operations of logical deduction.

At first sight, the mental imagery debate, with its roots in psychology, seems remote from problem solving. But a further consequence of the "maximal confusion" mentioned by Newell is that Simon did not hesitate to conflate phenomena usually seen as belonging to different realms. Take on the one hand the unconscious, fast and automatic processes of perception and memory, and on the other, the deliberate and (at least partially) conscious processes of mathematical problem solving. As general problems of information processing, Simon treated them as analogous, and tackled them through similar computer simulations.[26]

It is thus no surprise that, when tackling the imagery question, Simon (1978) went back to the way diagrams can help solve geometrical problems. Here is one of the examples he used to motivate his central insight. Let ABCD be a rectangle with AB twice as long as BC, let E be the midpoint of AB and F that of CD. Do the lines EF and AC intersect inside ABCD? For Simon, the sentence describing the problem is a first *representation* of it, from which one could laboriously solve it (given

[22] See footnote 10 above.

[23] Simon (1991: 205).

[24] Retrospective narratives often describe this conference as the founding of AI. For accounts of it by several participants, see McCorduck (2004: chap. 5).

[25] See McCorduck (2004: 126) and Simon (1991: 210).

[26] His overarching vision was broader still: he envisioned unified "sciences of the artificial" encompassing not just psychology and problem solving, but also, for instance, the study of organizations, with computer simulation as their central method (Simon 1996).

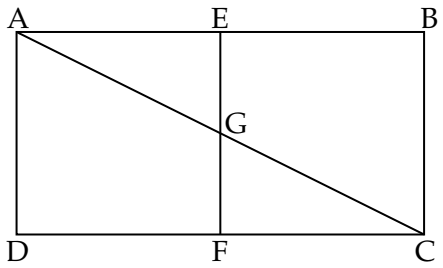

Figure 2: A geometric diagram which, according to Simon (1978: 6), allows solving an elementary problem more easily than equivalent sentences

an axiomatization of plane geometry). Instead, one could represent the data of the problem in other ways, say through cartesian equations, or more simply as Fig. 2.[27] According to Simon, these various representations differ not in their content—he called them "informationally equivalent"—but in the operations they facilitate. This is the idea I focus on below.

3 Introducing Simon's account

Before turning to the technical details, let me introduce Simon's computer model of representations informally, without presupposing familiarity with computer science.

Let us start with a concrete example that, though simple, suffices to illustrate Simon's main ideas. (This example is mine, not Simon's.) Imagine a university that lacks IT systems and keeps its full student files on paper in a basement. Each student receives an identification number when first enrolling, and student files are shelved in the basement ordered by student number. When a student enrolls, a new file is created and stored at the end of the last shelf in use. Later, whenever the student needs to deal with the university, they state their student number, allowing archivists to quickly locate and bring up their file from the basement. Everything runs smoothly in this imaginary university, save when a forgetful student loses their identification number: the archivists then have to go through all files one by one until they find the right one—an exceedingly tedious task.

One day, a new team takes the helm of our fictional university. To solve the issue of forgetful students, they decide to switch to a new system. From then on, the files are to be alphabetized by student name; whenever dealing with the university, students are simply to state their name. Few students being careless enough to forget their own name, this will spare archivists the chore of going through all files just to locate one. The main drawback of the new system is that it makes shelving a new

[27] Note that Simon is not worried, as mathematicians would be, about whether drawing the conclusion on the sole basis of a diagram is *legitimate*.

file more onerous: it is no longer enough to just add it at the end; one has to insert it at the right place according to the alphabetical order, which forces archivists to reorganize their shelves from time to time in order to make space.

With this example in mind, let us return to Simon. His first idea is that representations can be *compared* without having to deal with the question of what they are ultimately about:

> It is impossible to find an entirely neutral language in which to describe representations of information, for a language is itself a form of representation. We can avoid this difficulty, at least in part, by not attempting to describe representations directly, but by talking instead in terms of the equivalence of representations. [...] Specifically, I shall talk about *informational* equivalence and *computational* equivalence. [...] Two representations are informationally equivalent if the transformation from one to the other entails no loss of information, i.e., if each can be constructed from the other. [...] Two representations are computationally equivalent if the same information can be extracted from each (the same inferences drawn) with about the same amount of computation.[28]

In other words, any way we might devise to express what a representation means (for instance, by explaining it in natural language) would at bottom be nothing more than a translation into another form of representation; nevertheless, one can bypass this issue and talk of one representation containing the same, less or more information than another by focusing on whether one can be reconstructed from the other.[29]

For instance, our university's successive filing systems are *informationally* equivalent because one can convert the numerical system into the alphabetical one and vice-versa. On the other hand, they are patently not *computationally* equivalent in Simon's sense: some tasks that are slow in the old, number-based system (like finding a file from a student name) are quick in the new, alphabetical one, and conversely.

Of course, I am relying here on a vague conception of the amount of effort a task requires. But if we follow Simon and stay within a computer setting, this idea can be made rigorous: there is an area of computer science devoted to measuring the amount of computation that operations require, namely *computational complexity theory*. Its starting point is that in practice, what matters most is to know how the speed of a certain operation depends *on the size of the data*. Our fictional university

[28] Simon (1978: 4–5); his emphasis.

[29] This use of "information" is in the spirit of Shannon's "mathematical theory of communication" (which, indeed, is often called "information theory"): "The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have *meaning*; that is they refer to or are correlated according to some system with certain physical or conceptual entities. The semantic aspects of communication are irrelevant to the engineering problem." (Shannon 1948: 379, his emphasis). For a clear introduction, see Markowsky (2017).

reflects this well: in a university with just three students, any filing system would do; only when enrollment increases does it become necessary to distinguish between different systems. Computational complexity is meant to quantify such distinctions. Say our university uses a numerical filing scheme. Obtaining a file from a given number is then basically just as fast whether there are three students or thousands; accordingly, we say that this operation has a "constant" time complexity. On the other hand, obtaining a file by name might require going through every file to locate the right one, a task whose length is proportional to the number of students; the operation is thus said to have a "linear" time complexity. Notice that, as this constant-versus-linear contrast does not measure processing times directly but only up to a multiplicative factor, it is robust with respect to changes in the material implementation of the two systems (it will persist even if, e.g., archivists were replaced by robots doing the same tasks much faster).

Simon's second essential idea is that a representation can be described *abstractly*, independently of its physical realization. Let us go back once more to our fictional university. The administration, located on the university's upper floors, does not need to know precisely how student files are shelved in the basement. In fact, they only need to know whether student files should be requested by student number (as in the first system), or by name (as in the second)—in other words, what matters to them is knowing which requests will be fulfilled quickly. In fact, one could well imagine that the archiving service, disheartened by the idea of reorganizing tens of thousands of student files on a whim of upper management, would choose a trick. Instead of physically moving their files, they could keep relying on student numbers internally by creating a kind of alphabetical index that would list the number corresponding to each name. Indeed, the administration upstairs need not know whether, when they ask for Jane Doe's file (by name), the archivists go straight to the shelves, or whether they first look up Jane in their index; it does not matter to them as long as the file is on their desk a few minutes later (which the old system did not allow). In this scenario, one could say that the archivists are implementing an alphabetical system on the physical basis of a numerical one—while the powers that be are none the wiser.

In other words, for Simon, a "representation," thought of as a way of storing data, is *not* characterized by the details of its implementation but, more abstractly, by the requests it permits fulfilling quickly. Simon takes computers to have shown this decisively.[30]

How does this tie in with the imagery debate? If representational forms are defined by the fast operations they allow, then one can determine the forms of mental representations experimentally, by studying which tasks are slow or fast for experimental subjects. In a sense, this merely makes explicit the idea underlying psychological experiments like Shepard and Metzler's (see Section 1). They had

[30] "Modern computers," he writes, "have freed our thinking about representations." (Simon 1978: 9).

estimated the difficulty of a certain task (that of checking whether two drawings represented the same object) by measuring their experimental subjects' response time. Why should anything follow from this about the existence of mental images? The implicit justification is that only a mental representation "as images" can explain the specific distribution of response times observed—more precisely, the fact that the difficulty of the task depends on the angle between the two drawings. The relative difficulty of a certain operation is thus used as an *operational criterion* for recognizing mental images. In claiming that representational forms can be *defined*—and not merely *recognized*—operationally, Simon is just going a step further.

One question remains, and this is where Simon's work on problem solving comes in: if mental images are recognized or defined by certain operations, which operations should these be? It is probably fair to say, with critics of mental imagery, that the psychological literature often relied on a vague analogy with "external" images. Simon's peculiar methodology allowed him to make this analogy more precise. Remember that, at the "information-processing" level, Simon conflated mental processes and deliberate problem solving (possibly using diagrams), and simulated both using computer programs. Now take human problem-solving processes that rely on diagrams, and design adequate simulations of them. For Simon, the fast operations that define the data storage of such simulations will be constitutive of images in general, including mental ones.

This, then, is Simon's vision: introducing an abstract, computational definition of representational forms that applies equally to representations in the mind, in computers, or indeed on paper[31]—a definition which, among other things, would allow for a full empirical solution to the imagery debate.

4 "Data types": defining representations by an interface of operations

For Simon, different forms of representation are like different ways of storing data in a computer. This is more than an analogy: at the right level of analysis, he sees mental representations, diagrams on paper, and data stored in a computer as instances of one and the same abstract, computationally defined notion of representation. In this section and the next, I examine Simon's technical definitions of this notion, and argue that they basically amount to defining representations by the fast operations available on them.

Here is how Simon (1978) initially defined his notion of representation:

> Defining a representation means (1) specifying one or more data types,

[31] "Since we are concerned with representation at the symbolic or information-processing level, it will matter little whether the memory we are talking about resides in a human head or in a computer." (Simon 1978: 4)

> and (2) specifying the primitive (i.e., "fast") operations that can be performed on information in those data types.[32]

By the 1970s, in most programming languages, the pieces of data that could be manipulated by programs were separated into categories or "(data) types" (integers, floating-point numbers, characters, strings, arrays of integers, etc.). Accordingly, the most straightforward reading of clause (1) is as a requirement to specify the *nature* of the information being represented (e.g., in the case of our fictional archiving service, the specification that the data are made up of student files, each of which is made up of a name—i.e., a string of characters—a student number—i.e., an integer—a date of birth, grades, etc.). Clause (2) is less usual, and brings to mind the more sophisticated notion of an "abstract data type," which was being pioneered by computer scientists at the time. Simon may have had some awareness of their work, or may have been pursuing a similar train of thought independently.

Histories of computer science usually trace "abstract data types" back to the early 1970s, more precisely to papers by Tony Hoare and David Parnas, the latter a colleague of Simon at Carnegie Mellon.[33] The motivation of their work was to keep the growing complexity of software in check; to do this, their plan was to decompose complex programs into simpler components whose behavior could be specified independently of each other.[34] (Simon himself frequently discussed the advantages, for complex systems, of being composed of largely independent subsystems organized hierarchically.)[35] In the case of data, the important issue was to isolate the details of their storage from the rest of the program, and thus to codify explicitly and in advance the ways in which the program could interact with the data. Let us apply this idea to our university example: what matters then is to cleanly separate the administration from the archiving service, the latter being free to organize its shelves as it sees fit as long as it is able to fulfill the requests it receives—requests which, for this very reason, must be codified in advance. The first explicit definition of an abstract data type along such lines is in a 1974 paper by Barbara Liskov and Stephen Zilles:

> An *abstract data type* defines a class of abstract objects which is completely characterized by the operations available on those objects. This means that an abstract data type can be defined by defining the characterizing operations for that type.[36]

[32] Simon (1978: 7–8).

[33] In particular Parnas (1972a), Parnas (1972b) and Hoare (1972). See, e.g., Kutzler and Lichtenberger (1983: 1) and, for recent historical scholarship, Priestley (2011: chap. 10).

[34] The paradigmatic example of such a methodology is Edsjer Dijkstra's "structured programming." MacKenzie (2001: chap. 2) situates Dijktsra's and related research programs in a wider context; for a history richer in technical detail, see Priestley (2011: chap. 9–11). Parnas (2002) gives a clear retrospective statement of his motivations.

[35] See especially Simon (1962), reprinted in Simon (1969) and later editions (with revisions).

[36] Liskov and Zilles (1974: 51).

The "operations available" on an abstract data type are usually called its *interface*.

Note that Simon adds (inside parentheses and quotation marks) that the primitive operations making up the interface should be *fast*. The idea is that, *because* they are primitive, the operations in question will automatically be faster *comparatively* than other data operations: indeed, since other operations will have to be programmed out of the primitives, they will be slower than them. Think again of the archiving service of our university as if it was an abstract data type. In the spirit of a numerical filing scheme, assume we take as primitive operations that of accessing a file by student number and that of inserting a new file with a new student number. The operation of accessing a file by name, for instance, will then have to be programmed out of these, and hence will be much slower: presumably, one would examine the files corresponding to each student number one by one, calling the access-by-number primitive each time, until one finds a file having the right name.

This reading of Simon leads to a purely operations-based definition of representations, one that abstracts away completely from the physical organization of data. Such an account was made fully explicit by Simon's colleague John Anderson (1982), again with reference to (abstract) data types as used in computer science,[37] and is widespread today in cognitive psychology—a discipline which, thanks to its information-processing paradigm, is perhaps the truest contemporary heir to the original "cognitive science" research program (Núñez et al. 2019: 788). Simon's actual account, however, is somewhat less clear-cut than suggested so far—a complication we now need to clear up.

5 "Data structures": representations as abstract schemes of data organization?

Abstract data types, as presented in the previous section, offer a workable account of representations at the information-processing level. Simon himself, however, often seemed to conflate it with a different, more concrete account, based on what we would now call "data structures"—a phrase that Larkin and Simon (1987) used instead of "data types." I now present this other account and argue that, in the end, it fails as a viable alternative.

Recall that, in 1978, Simon defined representations as data types plus primitive operations. As discussed above, one could read the data types mentioned there as specifications of the nature of the data. In computer science terms, adding an interface of operations to such data types would yield an abstract data type. This would lead to a two-dimensional account of representations: they have a content that is shareable by other, "informationally equivalent" ones; their having a specific

[37] "It is possible [. . .] to define representations in terms of the processes that operate on them rather than the notations that express them. This is the point of view I want to adopt in this paper. It is the same idea as underlies the concept of data types in computer science." (Anderson 1982: 3).

"form" means *nothing else* than their having certain computational properties that can be captured as interfaces of primitive operations.

It seems, however, that Simon meant something different:

> A *data type* is some particular way of organizing information in memory. For example, among the data types that are commonly used in computing are lists and arrays. [...] The declaration that information will be represented in lists or in arrays does not say anything about the physical location of the information in memory.[38]

In our fictional university, a "data type" in this sense would not be merely about the contents of student files nor about the kinds of requests accepted by the archivists, but also about the archiving service's internal filing system—though without specifying it in all its physical details. Simon thus seems to be using the phrase in a sense closer to what computer scientists now call "data structures," of which the "lists" and "arrays" mentioned in the quote are simple examples; indeed, Larkin and Simon (1987) define representations as data structures (rather than data types) plus primitive operations. Representations would then be defined, not *only* by their interface of primitive operations, but *also* by some kind of abstract scheme of data organization.

To understand what Simon had in mind here, a detour through the practice of computer programming is helpful. Let us go through a simple example of data storage in a computer program, discussed by Donald Knuth in his classic programming textbook (which Simon referred to).[39] Imagine we need to work with an ordered list of elements (say student files or email messages). One may need to perform various operations on this list: access the first element, the last element, or in full generality the k^{th} element; insert or delete an element at the beginning, at the end, or in full generality at the k^{th} position; etc. Any way of organizing our list in computer memory will involve trade-offs, making some of these operations faster but others slower. However, most programs can be made to rely on some operations only, so that one can choose an organization in memory that optimizes these at the expense of the others.

For example, an ordered list can be used to implement a "queue," say a queue of emails to be sent. In this case, two operations are enough: adding a message at the end (behind the queue), and removing the message at the front to send it. For these operations to be performed efficiently, we need two things: a way to quickly find the first and last elements of the queue (that is, the message at the front—the next to be sent—and the one at the back, behind which new messages have to be inserted); and a way, for each element of the queue, to quickly locate the next one, so that whenever we send a message, we know which one will take its place at the front, that is, which one will become the next to be sent and deleted. In other words,

[38] Simon (1978: 8).

[39] See Knuth (1997: section 2.2.1). (Simon referred to the first, 1968 edition.)

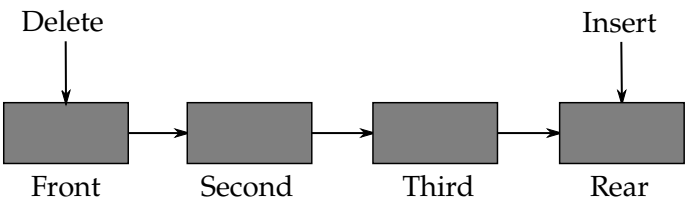

Figure 3: Representation of a queue structure, from Knuth (1997: 241)

the only *relations* between elements of the queue that need to be quickly accessed are those displayed on Fig. 3.

Importantly, a diagram like Fig. 3, while specifying what Knuth sometimes calls the "structural relationships"[40] relevant for our program, does not precisely lay out how the elements should be organized in memory: they can be arranged consecutively and in order, or far from one another and in disorder. For example, here is a common programming strategy that allows scattering the elements of a queue all over the computer memory. The idea is to supplement each element, say each email to be sent, with an additional piece of data, usually called a "pointer," indicating the *address in memory* of the next element in line (we also need to store pointers to the first and last elements). Whenever new emails come in, this setup allows storing them in any free space of the computer memory; it also makes it possible, as soon as an email has been sent further, to immediately free up its memory space.[41] There are other ways to implement a queue, however: a scheme in which emails are stored consecutively in a contiguous memory range would do just as well, as long as we have an efficient way of finding both ends of it. This is why Simon can speak of organizations in memory that do not specify "the physical location of the information." Such reasoning about the various ways to write an efficient program is common in the practice of computer programming, and Simon has even been a pioneer of it to some extent.[42]

The "queues" I have just described, like the "lists" and "arrays" that Simon mentions, are examples of what Knuth calls "information structures," usually named "data structures" today. These are less abstract than data types (which only specify the allowable operations) but are still an abstraction from the details of the program, which would lay out more precisely how data are to be organized in memory. This seems to be what Simon had in mind. Yet, as we shall see, it cannot provide him

[40] Knuth (1997: 232); he also speaks of "structural information" (see below, footnote 43).

[41] What we have just described is usually called a "chained list" (see Knuth 1997: §2.2.1).

[42] In the 1950s already, Newell, Shaw, and Simon had designed—for their *Logic Theory Machine*—the first programming language having data manipulation primitives, based on lists, that were far removed from the physical organization of the underlying computers. Their work had a significant impact on the later history of programming languages, especially on what we now call "functional" programming (see Priestley 2017).

with the abstract account of representation he needs.

The fundamental problem with this new account is that the "form" of a representation is split into *two levels* whose relationship is unclear. There is a data structure, which seemingly provides an abstract data-organization scheme, and there are primitive operations; Simon cannot abandon the latter altogether, since only through them can a given form of mental representation (say mental images) be recognized experimentally. The difficulty consists in clarifying how these two levels are related. Anderson (1978) famously argued that it is precisely the pervasiveness of two-level accounts of representations, together with a lack of clarity as to how one level could constrain the other, that made the imagery debate so intractable. Indeed, if the two levels were unrelated, there would be no hope of recognizing mental images merely by the operations experimental subjects can perform quickly. Generally, if one were free to postulate any fast operation independently of the way data are organized, then it would be impossible to fully distinguish forms of representation just from the fast operations on them.

Accordingly, while Simon initially presented the two levels as separate ingredients making up a representation, he immediately added:

> [T]he declaration of a data type means that certain *primitive operations* are available for accessing, storing, and modifying data. A list representation requires a primitive operation of *find.next*, which, given a particular symbol as its argument, will access the succeeding symbol in the list to which the first belongs.[43]

This seems to suggest that a data-organization scheme would *determine* the relevant primitive operations.

Unfortunately, there seems to be no way to make sense of such an intrinsic link between data organization and fast operations without abandoning Simon's initial ambitions. At the heart of the issue is the interpretation of diagrams such as Fig. 3, which Simon (1978) uses to explain what his "data types" (our "data structures") are. Following Knuth, I wrote above, in vague terms, that the arrows of Fig. 3 indicate those "structural relationships" that are relevant for a computer program using a queue. What exactly does this mean?

A straightforward way to interpret the arrows, suggested by Simon's remark about *find.next*, is as operations that any implementation of a queue should make efficient. A queue would then be characterized merely by the fact that one can quickly access the queue's first and last elements, as well as the successor of any particular element. But then, Simon's abstract models of data organization would be nothing more than a roundabout way of stipulating which operations should be fast: his data-organization account would collapse back onto a fast-operations one.

Alternatively, it is tempting to say that the queue diagram indicates those pieces

[43] Simon (1978: 8); his emphases.

of information that should be encoded "explicitly."[44] But what does it mean for a piece of information to be "explicitly" represented? Kirsh (1990) convincingly argued that this is not clear at all; in fact, in a standard computer setting, his explication of explicitness basically reduces it to accessibility in constant time.[45] Once again, we fall back on a fast-operations account.

There are other, perhaps more plausible ways of understanding Simon's explanation of data structures, but crucially, they end up tying his account of representational forms to *a particular computational architecture*, thus making it impossible to treat minds, diagrams on paper, and computers simultaneously. For instance, one could interpret the arrows of Fig. 3 concretely as indicating that the source of the arrow should contain a pointer to the memory address of the target of the arrow. But this presupposes that each piece of data has a memory address and that accessing it from its address is fast, which is only true in specific material architectures. Switching to computers with a different architecture, for instance with memory slots that are accessed by content rather than by address ("content-addressable memory"), would change the relationship between data organization and fast operations. In other words, making the definition of a data structure concrete (as textbooks usually do)[46] robs it of the independence from physical realization that Simon needs.

In sum, an account based on data structures either falls back on the abstract, purely operations-based account presented in the previous section, or is too concrete to accomplish Simon's goals. However that may be, carefully distinguishing both accounts did not, in practice, matter much to Simon. In his simulations, he constantly used data structures built on top of list structures, which were easily implemented on a standard computer architecture, and which he believed to be plausibly realizable by neurons;[47] and he apparently thought that for all intents and purposes, any representation in the abstract sense—any set of primitive operations—could be modeled well enough using such (concrete) data structures and suitable operations on them. In other words, even if he viewed representations, at the information-processing level, as nothing but abstract data types, he saw no problem with systematically using list-based data structures as serviceable stand-ins for them. This is how he proceeded, together with Jill Larkin, in his work on the use of diagrams in scientific problem solving, which we are now in a position to assess.

[44] Larkin and Simon (1987) sometimes use this word (e.g., p. 67); similarly, Knuth (1997: 238) uses the word "directly": "Data usually has much more structural information than we actually want to represent *directly* in a computer [. . .]" (my emphasis).

[45] See also Clark (1992).

[46] For instance, Knuth (1997) locates his entire discussion in the context of computers with memory addresses, going so far as to introduce a particular machine language from the outset.

[47] See, e.g., Simon (1989: 383–384).

6 External representations in scientific practice

Let us finally turn to Larkin and Simon (1987). Their paper is about external representations used in scientific problem solving (especially diagrams), rather than mental representations. For Simon, who systematically equated the two cases, this made no difference; moreover, mental images were still his main target, as he later explained:[48]

> In order to deal with the difficulties one by one, [Jill Larkin and I] fudged a bit, alleging that we were talking about diagrams on paper rather than mental pictures; but most of our argument carries over in a straightforward way.[49]

Jill Larkin, however, was a psychologist specialized in science education. She was primarily interested, not in mental images, but in scientific reasoning with external representations,[50] and this is what the paper is actually about.

So, what does the operations-based account of representations presented above amount to in this new context? As hinted at in the introduction, external representations as used in scientific practice would normally be conceived as artifacts that are individuated, and made meaningful, by their belonging to some kind of representational system. But as we shall see, representations in this usual sense cannot, in and of themselves, be compared computationally *à la* Simon; this is only possible relative to a particular way of using them, which determines the role they play in a broader problem-solving process—a role which, as explained in the two previous sections, should be understood as an "abstract data type." A diagram and a table, for instance, are obviously different representations in the usual sense. Depending on how they are used, however, Simon's computational perspective can actually treat them as if they were the same. A close reading of Larkin and Simon's examples will make this clear.

Larkin and Simon's first example is a statics problem about weights and pulleys, which they first state sententially:

> We have three pulleys, two weights, and some ropes, arranged as follows: The first weight is suspended from the left end of a rope over Pulley *A*. The right end of this rope is attached to, and partially supports, the second weight. Pulley *A* is suspended from the left end of a rope that runs over Pulley *B*, and under Pulley *C*. [. . .][51]

Our authors formalize these sentences in a Lisp-like programming language to get a first data structure, which looks like this:

[48] Simon kept working on visual reasoning throughout the 1990s; for representative samples, see Qin and Simon (1995) and Tabachneck-Shijf, Leonardo and Simon (1997).

[49] Simon (1989: 383).

[50] See for example Larkin (1983; 1989).

[51] Larkin and Simon (1987: 72).

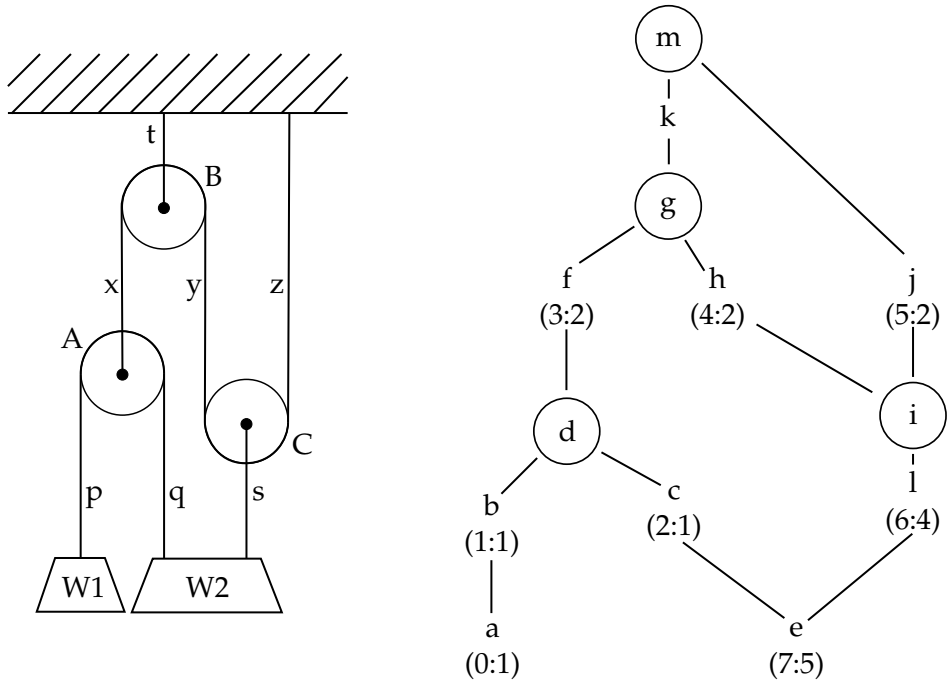

Figure 4: Diagrams of: a statics problem (left); the data structure Larkin and Simon use to simulate a diagram-based method to solve it (right) (from Larkin and Simon 1987: 73, 79)

```
(Weight W1) (Rope Rp) (Rope Rq) (Pulley Pa)
(hangs W1 from Rp)
(pulley-system Rp Pa Rq) [...]
```

They also state, and formalize, a number of rules with which one can derive new information (e.g., a rule encoding the fact that the two sides of a rope running over a pulley have equal tension).[52] Given this setup, here is a naive solution method: for each rule, systematically try applying it to all possible sentences in the data structure, and do this over and over again. The issue with this method is its inefficiency: at each stage, one may need to go through everything one knows multiple times before finding sentences to which a rule can be applied.

Larkin and Simon's idea is that a diagram (as on the left-hand side of Fig. 4) allows for a more efficient solution because it can guide one to the right applications of the rules, avoiding a costly exhaustive search. For instance, if one knows the value of weight *W1*, one can deduce the tension of rope *p*, then that of rope *q*, and so on, always moving from an element to another that is connected to it in the diagram. This, according to our authors, is how human problem solvers would normally proceed.

[52] Larkin and Simon (1987: 74).

To simulate this human, diagram-based method, Larkin and Simon use a data structure that is organized by localization (rather than sequentially). Here is how it works. Everything known about, say, pulley A is gathered in a single slot in memory; this slot is connected to three other memory slots, those containing everything known about the elements that have a direct physical connection to A, namely ropes p, q and x. To clarify the organization of their data structure, Larkin and Simon represent it as on the right-hand side of Fig. 4. (Note that this diagram merely *displays* the underlying data structure, which is not *itself* a diagram on paper.) The solution algorithm then works as follows: whenever one has used a rule to deduce something new at a given memory slot, one searches the slots linked to it for further opportunities to apply rules.

We now get to the crucial points. First, in what sense do the "data structure" and operations of our authors' program correspond to the diagram on paper and to the operations performed with it by human problem solvers? Simon would presumably say that, at the information-processing level, they play the same role: as abstract representations, they are the same. Our analysis above of Simon's abstract concept of representation allows us to clarify what this amounts to. It merely means that they make the same (comparatively) fast operations available to the rest of the problem-solving process; whatever our authors might appear to suggest, there is no theoretical justification for taking the data structure to correspond to the diagram in any deeper way. In other words, the core of Larkin and Simon's method is to speculate on what the diagram allows a human being engaged in a particular problem-solving process to do quickly, then design a data structure that allows the corresponding operations to be performed efficiently.

Second, it then becomes clear that Larkin and Simon's method can only capture *particular uses* of representations. In the preceding example, their "diagrammatic" data structure is designed to make it quick, given some element (say a pulley), to locate the elements immediately connected to it (say various ropes)—but nothing more. However, this limited computational role could be played by other sorts of representations, like a graph or a table. Conversely, human beings might use the diagram itself for various other purposes: to quickly recognize symmetries, to locate the bottommost weight, and so on. In other words, when Larkin and Simon speak of modeling "the diagram," they are only modeling a particular computational role of it—one that could be played by other representations and is only incidentally connected, in the limited context of a particular problem-solving process, to the diagram as we usually understand it.

Larkin and Simon's second example is a geometrical problem that is easily solved using a few elementary theorems. They initially phrase the problem sententially in terms of *points* and *lines*. However, the theorems one needs for a solution—among others, that alternate interior angles are equal—are standardly phrased in terms of *segments*, *angles*, and *triangles*. So, to solve the problem from its sentential formulation, one first needs to identify the segments, angles, and triangles present

in the configuration. This is precisely, Larkin and Simon claim, what the diagram makes easy: these elements, they write, "are perceptually obvious to any educated person looking at [the] diagram."[53]

This second example reinforces the two points made above. First, their purportedly diagrammatic data structure only corresponds to the diagram on paper in a shallow way—namely, in that both have similar fast operations available on them. A peculiar feature of their treatment of this case makes this particularly clear. One would expect Larkin and Simon to construct some kind of list structure corresponding to the diagram's points and lines, structured in such a way that it lends itself to, say, a fast angle-recognition operation—like the diagram apparently does for the human visual system. But they fail to do this. Instead, they set up a *computationally costly* search process that produces, from their initial data structure made up of points and lines, a "perceptually enhanced" one containing angles and the like—and then take this perceptually enhanced data structure to directly correspond to the diagram as used by human problem solvers, assuming that the relevant perceptual elements are produced "for free" by "the eye and the diagram."[54] This strategy might look like cheating, but it makes sense if one defines the representation as a black box (i.e., as an abstract data type), the inner workings of which are implemented inefficiently by Larkin and Simon's simulation, but presumably very efficiently by the diagram on paper together with a human visual system.

Second, as Larkin and Simon freely admit, while *angles* are immediately available in the "enhanced data structure" meant to capture the diagram's advantages, *alternate interior angles* are not—even though agents with significant training in geometry see them easily. "Although we could add patterns like alternate interior angles," they write, "it is not clear that such patterns are obvious to everyone, and we will develop our program to reason about them explicitly."[55] This shows how sensitive their account is to the way a representation is used: even when using the *same* geometrical diagram to solve the *same* problem, different agents, depending on their expertise, might actually be using it in different ways that would be best captured using different "data structures."

In sum, strictly speaking, one cannot say that the diagram and the equivalent sentences, in and of themselves, are "computationally different"—such a comparison is only meaningful relative to a particular way of using them. Nevertheless, one might still be tempted to compare them computationally in a broader sense: is it not true that, given the visual abilities of human beings, the diagram *lends itself* to various advantageous uses in a way the sentences do not? Perhaps, but the link between representational form and potential efficient uses is not straightforward. One would need something like a high-level account of the sorts of patterns the human visual system may be trained to recognize efficiently, a tall order; and such

[53] Larkin and Simon (1987: 88).
[54] Larkin and Simon (1987: 92).
[55] Larkin and Simon (1987: 88).

efficient recognition would still require appropriate training. One should thus be wary of seeing computational advantages as intrinsic to representations in the usual sense of the word; use plays an irreducible role.

Conclusion

Simon started from a computer model, that of data storage in a computer, in which the concepts of informational and computational equivalence made clear sense; he then undertook to treat mental as well as "external" representations (in particular scientific representations such as diagrams drawn on paper) through this computer model (Sections 1–2). His move from the computer setting to representations in general could be justified in two ways. First, it relied on a broad postulate which grounded all of Simon's work, namely that at the right level of analysis—the "functional" or "information-processing" level—every problem-solving process could be identified with a computer program. Second and more specifically, Simon articulated an abstract notion of representation that made sense at this level of analysis. This abstract notion was meant to legitimize his approach by showing that at the information-processing level, external representations, mental representations and data storage in a computer could indeed be identified.

As we have seen, Simon's abstract account amounts to nothing more than defining representations by an interface of fast operations available on them, disregarding any question of physical realization (Section 4)—although his systematic use of list-based data structures in his problem-solving simulations sometimes muddies the waters (Section 5).

In experimental psychology, this is a plausible—and now widespread—move, as long as representations are only studied by measuring response times or otherwise observing behavior. But in the case of external representations, as studied by Larkin and Simon (Section 6), Simon's abstract account clashes with more standard approaches that see representations as meaningful artifacts, to the point that Simon's approach ends up shifting the topic.

In the end, instead of fully reducing representational differences (in the standard sense) to computational differences, Simon's computational approach is only able to compare representations relative to a particular way of using them. One and the same representation in the standard sense can correspond to different representations in Simon's abstract sense, if they are used in different ways; for example, Larkin and Simon's geometrical diagram when used by a novice, and the same diagram when used by an expert, will correspond to different representations in Simon's sense. Conversely, two representations that are patently different in the usual sense, say a diagram and a table, can be used in computationally equivalent ways and hence be the same representation in Simon's sense. This means that, despite what Simon initially meant to do—and despite what his slogan might still suggest—representational differences cannot be accounted for in a purely computational way.

Acknowledgments. For comments and discussion, I would like to thank my anonymous reviewers as well as Jeremy Avigad, Gianni Gastaldi, Valeria Giardino, Yacin Hamami, Nicolas Michel, John Mumma, Dirk Schlimm, Gisele Secco, and Henri Stephanou. I would also like to thank the audience at HaPoC 2021, and in particular Simone Martini, for insightful remarks.

References

Anderson, John R. (1978). "Arguments Concerning Representations for Mental Imagery". In: *Psychological Review* 85.4, pp. 249–277.

— (1982). *Representational Types: A Tricode Proposal*. Technical Report 82-1. Washington, D.C.: Office of Naval Research, June 10, 1982. URL: https://apps.dtic.mil/dtic/tr/fulltext/u2/a116887.pdf.

Clark, Andy (1992). "The Presence of a Symbol". In: *Connection Science* 4.3–4: *Representation, Development and Situated Connectionism*, pp. 193–205.

Crevier, Daniel (1993). *AI. The Tumultuous History of the Search for Artificial Intelligence*. New York: Basic Books.

Crowther-Heyck, Hunter (2005). *Herbert A. Simon. The Bounds of Reason in Modern America*. Baltimore and London: The Johns Hopkins University Press.

Dick, Stephanie (2015). "Of Models and Machines: Implementing Bounded Rationality". In: *Isis* 106.3, pp. 623–634.

Gelernter, Herbert (1963). "Realization of a Geometry-Theorem Proving Machine". In: *Computers and Thought*. Ed. by Edward A. Feigenbaum and Julian Feldman. New York: McGraw-Hill, pp. 134–152. Repr. from *Information Processing*. Paris, Munich, and London: UNESCO, Oldenbourg, and Butterworths, pp. 273–282.

Goodman, Nelson (1968). *Languages of Art. An Approach to a Theory of Symbols*. Indianapolis: Bobbs-Merrill.

Haugeland, John (1978). "The Nature and Plausibility of Cognitivism". In: *Behavioral and Brain Sciences* 1.2, pp. 215–260.

— ed. (1981). *Mind Design. Philosophy, Psychology, Artificial Intelligence*. Cambridge, Mass. and London: Bradford Books—MIT Press.

— (1998). "Representational Genera". In: *Having Thought. Essays in the metaphysics of mind*. Cambridge, Mass. and London: Harvard University Press, pp. 171–206. Repr. from *Philosophy and Connectionist Theory*. Ed. by William Ramsey, Stephen Stich, and David Rumelhart. Hillsdale, New Jersey: Lawrence Erlbaum Associates, pp. 61–89.

Hoare, C. A. R. (1972). "Proof of Correctness of Data Representations". In: *Acta Informatica* 1.4, pp. 271–281.

Humphreys, Paul (2004). *Extending Ourselves. Computational Science, Empiricism, and Scientific Method*. Oxford and New York: Oxford University Press.

Kirsh, David (1990). "When Is Information Explicitly Represented?" In: *Information, Language and Cognition*. Ed. by Philip P. Hanson. Vancouver Studies in Cognitive Science 1. Vancouver: University of British Columbia Press, pp. 340–365.

Knuth, Donald E. (1997). *The Art of Computer Programming*. *Fundamental Algorithms*. 3rd ed. Reading, Mass.: Addison Wesley. 1st ed. 1968.

Kosslyn, Stephen M. (1980). *Image and Mind*. Harvard University Press: Cambridge, Mass. and London.

Kulvicki, John (2010). "Knowing with Images: Medium and Message". In: *Philosophy of Science* 77.2, pp. 295–313.

Kutzler, B. and F. Lichtenberger (1983). *Bibliography on Abstract Data Types*. Informatik-Fachberichte 68. Berlin and Heidelberg: Springer-Verlag.

Larkin, Jill H. (1983). "The Role of Problem Representation in Physics". In: *Mental Models*. Ed. by Dedre Gentner and Albert L. Stevens. Hillsdale, New Jersey: Lawrence Erlbaum Associates, pp. 75–98.

— (1989). "Display-Based Problem Solving". In: *Complex Information Processing. The Impact of Herbert A. Simon*. 21st Carnegie-Mellon Symposium on Cognition. Ed. by David Klahr and Kenneth Kotovsky. Hillsdale, New Jersey: Lawrence Erlbaum Associates, pp. 319–341.

Larkin, Jill H. and Herbert A. Simon (1987). "Why a Diagram Is (Sometimes) Worth Ten Thousand Words". In: *Cognitive science* 11.1, pp. 65–100.

Liskov, Barbara and Stephen Zilles (1974). "Programming with Abstract Data Types". In: *ACM SIGPLAN Notices* 9.4, pp. 50–59.

MacKenzie, Donald (2001). *Mechanizing Proof. Computing, Risk, and Trust*. Inside Technology. Cambridge, Mass. and London: MIT Press.

Markowsky, George (2017). *Information Theory*. In: *Encyclopædia Britannica*. June 16, 2017. URL: https://www.britannica.com/science/information-theory.

McCorduck, Pamela (2004). *Machines Who Think. A Personal Inquiry into the History and Prospects of Artificial Intelligence*. 2nd ed. Natick, MA: A K Peters. 1st ed. New York: W. H. Freeman, 1979.

Mirowski, Philip (2002). *Machine Dreams. Economics Becomes a Cyborg Science*. Cambridge and New York: Cambridge University Press.

Newell, Allen, J. Clifford Shaw, and Herbert A. Simon (1963). "Chess Playing Programs and the Problem of Complexity". In: *Computers and Thought*. Ed. by Edward A. Feigenbaum and Julian Feldman. New York: McGraw-Hill, pp. 39–70. Repr. from *IBM Journal of Research and Development* 2.4, pp. 320–335.

Newell, Allen and Herbert A. Simon (1956). "The Logic Theory Machine. A Complex Information Processing System". In: *IRE Transactions on Information Theory* 2.3, pp. 61–79.

Newell, Allen and Herbert A. Simon (1963). "Empirical Explorations with the Logic Theory Machine". In: *Computers and Thought*. Ed. by Edward A. Feigenbaum and Julian Feldman. New York: McGraw-Hill, pp. 109–133. Repr. from *Proceedings of the Western Joint Computer Conference*. New York: Institute of Radio Engineers, pp. 218–230.

— (1972). *Human Problem Solving*. Englewood Cliffs, N.J.: Prentice-Hall.

— (1976). "Computer Science as Empirical Inquiry. Symbols and Search". In: *Communications of the ACM* 19.3, pp. 113–126.

Núñez, Rafael et al. (2019). "What happened to cognitive science?" In: *Nature human behaviour* 3, pp. 782–791.

Parnas, David L. (1972a). "A Technique for Software Module Specification with Examples". In: *Communications of the ACM* 15.5, pp. 330–336.

— (1972b). "On the Criteria to Be Used in Decomposing Systems into Modules". In: *Communications of the ACM* 5.12, pp. 1053–1058.

— (2002). "The Secret History of Information Hiding". In: *Software Pioneers. Contributions to Software Engineering*. Ed. by Manfred Broy and Ernst Denert. Berlin and Heidelberg: Springer, pp. 399–409.

Priestley, Mark (2011). *A Science of Operations. Machines, Logic and the Invention of Programming*. History of Computing. London and Dordrecht: Springer.

— (2017). "AI and the Origins of the Functional Programming Language Style". In: *Minds & Machines* 27.3, pp. 449–472.

Pylyshyn, Zenon W. (1973). "What the Mind's Eye Tells the Mind's Brain. A Critique of Mental Imagery". In: *Psychological Bulletin* 80.1, pp. 1–24.

Qin, Yulin and Herbert A. Simon (1995). "Imagery and Mental Models". In: *Diagrammatic Reasoning. Cognitive and Computational Perspectives*. Ed. by T. I. Glasgow,

N. H. Narayanan, and B. Chandrasekaran. With a forew. by Herbert A. Simon. Menlo Park and Cambridge: AAAI Press and MIT Press, pp. 403–434.

Shannon, Claude E. (1948). "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3, pp. 379–423.

Shepard, Roger N. and Jacqueline Metzler (1971). "Mental Rotation of Three-Dimensional Objects". In: *Science*. New ser. 171.3972, pp. 701–703.

Simon, Herbert A. (1947). *Administrative Behavior. A Study of Decision-Making Processes in Administrative Organization*. With a forew. by Chester I. Barnard. London and New York: The Free Press—Macmillan. 2nd ed. 1957; 3rd ed. = Simon 1976; 4th ed. 1997.

Simon, Herbert A. (1962). "The Architecture of Complexity". In: *Proceedings of the American Philosophical Society* 106.6, pp. 467–482. Repr. in Simon 1969, chap. 4 (2nd ed., chap. 7; 3rd ed. = Simon 1996, chap. 8).

— (1969). *The Sciences of the Artificial*. Cambridge, Mass. and London: MIT Press. 2nd ed. 1981; 3rd ed. = Simon 1996.

— (1972). "What Is Visual Imagery? An Information Processing Interpretation". In: *Cognition in Learning and Memory*. Ed. by Lee W. Gregg. New York et al.: Wiley, pp. 183–204.

— (1976). *Administrative Behavior. A Study of Decision-Making Processes in Administrative Organization*. With a forew. by Chester I. Barnard. 3rd ed. London and New York: The Free Press—Macmillan.

— (1978). "On the Forms of Mental Representation". In: *Perception and Cognition. Issues in the Foundations of Psychology*. Ed. by C. Wade Savage. Minnesota Studies in the Philosophy of Science IX. Minneapolis: University of Minnesota Press, pp. 3–18.

— (1989). "The Scientist as Problem Solver". In: *Complex Information Processing. The Impact of Herbert A. Simon*. 21st Carnegie-Mellon Symposium on Cognition. Ed. by David Klahr and Kenneth Kotovsky. Hillsdale, New Jersey: Lawrence Erlbaum Associates, pp. 375–398.

— (1991). *Models of My Life*. The Alfred P. Sloan Foundation series. New York: Basic Books.

Simon, Herbert A. (1996). *The Sciences of the Artificial*. 3rd ed. Cambridge, Mass. and London: MIT Press.

Tabachneck-Schijf, Hermina J. M., Anthony M. Leonardo, and Herbert A. Simon (1997). "CaMeRa: A Computational Model of Multiple Representations". In: *Cognitive Science* 21.3, pp. 305–350.

Thomas, Nigel J.T. (2014). *Mental Imagery*. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Stanford: Metaphysics Research Lab, Stanford University, Aut. 2014. URL: https://plato.stanford.edu/archives/fall2014/entries/mental-imagery/.

Vorms, Marion (2011). "Representing with Imaginary Models: Formats Matter". In: *Studies in History and Philosophy of Science Part A* 42.2, pp. 287–295.

— (2012). "Formats of Representation in Scientific Theorizing". In: *Models, Simulations, and Representations*. Ed. by Paul Humphreys and Cyrille Imbert. Routledge Studies in the Philosophy of Science 9. New York and Abingdon: Routledge, pp. 250–269.