Increasing returns and the efficient organization of information*

Michael Mandler†

Royal Holloway College, University of London

This version: December 2020

Abstract

Increasing returns are inherent in information: the number of possible facts that can be organized is an exponential function of the number of times the information is sorted. Just as factories should run at a large scale in the face of increasing returns, information should be sorted into a small number of categories so that the number of sortings can be large. We provide two applications. First, each device in an information storage array should attain 2 states. By explaining the prevalence of transistors, this result resolves a puzzle of the existing literature which advocates 3-state devices. Second, questions designed to elicit information should have a small number of answers each though usually more than 2. The problems are formally similar but an additional cost of answering questions counterintuitively raises the efficient number of answers: agents typically fail to deduce that if all but one of the answers are rejected then the remaining answer must be accepted.

JEL codes: D91, D83, C61

Keywords: increasing returns, information storage, information elicitation, bits, ternary trees.

*For helpful discussions and suggestions, I thank Francesc Dilmé, Itzhak Gilboa, Kim Kaivanto, R. Vijay Krishna, and Michael Richter.

†Address: Department of Economics, Royal Holloway College, University of London, Egham, Surrey TW20 0EX, United Kingdom. Email: m.mandler@rhul.ac.uk

1 Introduction

Increasing returns are inherent in the organization of information. Suppose that information can be stored or organized in devices that can attain k states each, for example, in transistors where $k = 2$. Each value of k defines a different type of information 'factory' and the number N of k-state devices in use indicates the scale of the factory. The number of configurations of the devices in a k-state factory of size N and hence the number of facts the factory can store is k^N, an exponential function of the number of devices. A strong form of increasing returns obtains.

Just as in a conventional industry, increasing returns implies it will be efficient to operate an information factory at a large scale. For example, when storing a given number of facts – the output target – it will under mild assumptions be more efficient to operate a factory at a large scale with devices where k is small than a factory at a small scale where k is large. The efficient storage of information will consequently drive k to its minimum value of 2, which explains the dominance of transistors in actual practice.

This static case of increasing returns will serve as a preliminary for my main topic, the sequential organization of information by human agents. Suppose an individual can acquire information over time that partitions the possible states of nature into progressively finer grids. If for example the state of nature specifies the weather next week then each day's weather report will further narrow down the states that remain possible. Let the first round of information sort states of nature into k_1 categories, the second round sort the states in each first-round category into k_2 further categories, and so on. The number of distinct events identified after N rounds of sorting will then be $k_1 \times \cdots \times k_N$, an exponential function of the geometric mean of the k_i. Strong increasing returns again obtains and it will therefore be efficient for the k_i to be small and for the scale of operation N to be large (many rounds of sorting).

In both the static and sequential organization of information, these small k conclusions will depend in part on the cost of devices or of sorting. In information storage, the devices that encode information record deviations from a default state, e.g., by letting a 1 replace a 0 in a device that stores a bit. The number of actively recordable states in a k-state

device is thus $k-1$. A 1-state device for example is inert and should be free: its state can never change and it can store no information. One plausible hypothesis is therefore that a k-state device has a cost proportional to $k-1$. We will show that two-state devices then always store one out of n facts more efficiently than devices with $k \neq 2$, for any n. This conclusion holds with great generality: 2-state devices are optimal even when the marginal cost of adding a further state to a k-state device diminishes to 0 as k increases rather than remaining constant. It is therefore no accident that actual information storage devices – transistors – always attain two states.

An engineering literature on 'radix economy' has explored how many states information storage devices should attain and has come to a different conclusion: efficient devices should attain three states.[1] This result however rests on an assumption that k-state devices have a cost proportional to k, which is hard to justify. Since 1-state devices will never be used, costs proportional to k impose a marginal cost for 2-state devices that is twice the marginal cost of 3-state devices. Without this handicap, 2-state devices will dominate. The engineering literature has not dwelled on the mismatch between its normative advice and the actual storage of information. For economists, the mismatch is more troubling and it will disappear in the optimization theory I lay out.

Although the engineering argument for $k = 3$ devices is not well-suited to its original setting, it turns out to be the key to the dynamic problem of how people should organize information sequentially. I will consider, for concreteness, an agent that has some information we need to elicit. For example, a doctor might need to diagnose a patient who knows the state of nature, a long list of details about his or her medical condition. To extract the relevant facts, the agent is guided through a tree of questions; the goal is to minimize costs as measured by the time the agent spends reading or processing the answers to questions. For each question, the answers sort the states of nature that remain possible into categories and each path of answers therefore narrows down the event where the state of nature lies until a diagnosis can be made. If along each path the first question has k_1 possible answers, the second k_2 answers, and so on, then we arrive at the $k_1 \times \cdots \times k_N$ formula for the number of events given earlier. Trees of full generality are

[1] The founding work is Engineering Research Associates (1950). I review later developments in section 2.

covered in section 3.

Since increasing returns are present, large values for the k_i can be efficiently replaced by small values. But which small values? The information elicitation problem can sometimes nearly repeat the information storage problem and have a similar solution. When an agent facing a question with k_i possible answers has read and rejected $k_i - 1$ of the answers then he or she may be able to deduce that the final answer must be correct – just as in a k-state storage device the absence of any of the $k - 1$ deviations from the default state implies that the default state must obtain. In the words of Sherlock Holmes, 'when you have eliminated the impossible, whatever remains, however improbable, must be the truth.'[2] Such an agent takes time proportional to $k_i - 1$ to answer a k_i-answer question and the quickest tree will then ask binary (two-answer) questions, comparably to efficient machine storage.

But a cost of processing answers proportional to $k_i - 1$ will often be implausible. While people can occasionally deduce an event from the negation of its complement, they often fall short of the Holmesian ideal: an agent who is asked a question will typically need to read or hear the content of all k_i answers. The first $k_i - 1$ answers might not fully reveal how the questioner conceives of the state space or how states of nature map to answers. The first two of three categories for an ailment, say 'physical' and 'psychological', will not pin down the content of the third category: can the first two be somehow combined or is there a third possibility? The time it takes an agent to answer a question can therefore be proportional to k_i, the same assumption the engineering literature has advocated for the cost of information storage. The efficiency of binary questions is then overturned. Depending on the exact objective function and on the number of possible diagnoses, the optimal number of answers to questions can vary, with the engineering solution of 3 answers per question often predominating.

In sum, increasing returns implies that information should be organized coarsely, but just how coarsely depends on the details. If moreover the cost of answering questions rises from $k_i - 1$ to k_i efficient questions will end up with more answers rather than fewer. Though counterintuitive, this result gives a better fit with the evidence: machines should

[2] Conan Doyle (1890).

and do use bits while the questions people face should be and frequently are nonbinary.

The sequential elicitation problem I consider is closely related to two previously studied optimization problems. The first is the minimization problem that defines informational entropy and underlies the theory of rational inattention, the subject of considerable research since Sims (2003) (see Maćkowiak et al. (2018)). If to discover which of n states obtains we can ask a tree of questions with k answers each then the minimum expected number of questions we must ask is well-approximated by the entropy $H(\pi) = -\mathbb{E}\log_k \pi$ where $\pi(i)$ is the probability of state i: the minimum expected number of questions must lie between $H(\pi)$ and $H(\pi) + 1$ (see, e.g., Aigner (2007)). A standard entropy problem, such as Huffman (1952), takes n, k, and π as given and finds the optimal tree that achieves this minimum. This paper also considers the time it takes to proceed through a tree of questions but instead optimizes with respect to k, usually an uninteresting scaling factor chosen to measure information in the same units as one's communications channel. If we simply minimized the number of questions an agent will face with respect to k, we would arrive at the vacuous conclusion that the communications channel should be as wide as needed: by setting $k = n$ we can identify the state with a single question. The problem becomes interesting only when the model incorporates the costs imposed on agents by the number of answers that a question can have. Accordingly, I will weight questions by their numbers of answers. Given the link between entropy and rational inattention, our results help explain how much information agents should acquire when processing answers is costly.

The sequential elicitation problem is also closely tied to the Radner (1993) analysis of decentralized information processing in organizations. A Radner organization can also be seen as a tree: agents at the bottom receive and process information and hand their results to higher-ups who further process the information and hand their results to their superiors, and so on, until the top agent can announce the organization's action or conclusion. Since the flow of information is decentralized or bottom-up, parallel processing across levels of the organization is efficient; otherwise some higher-ups would remain idle. Our problem in contrast is top-down and processing is therefore serial rather than parallel: the agent must answer the first question before turning to the second question. The

parallel vs. serial distinction however tends to disappear once organizations face a flow of problems, as Bolton and Dawatripont (1994) and Radner himself emphasized, since higher-ups can then spend their time finishing the processing of previous problems rather than remaining idle. One can therefore interpret the minimum time it takes an agent to answer a tree of questions in this paper as the minimum time an organization needs to process information. Radner (1993) also showed that the returns to scale in information processing will depend subtly on how returns are defined; the approach I take singles out the unambiguous increasing returns effect on one feature of the optimal organization of information, namely the number of answers to questions or states per device.

The optimization theory in this paper sheds light on the origins and consequences of bounded rationality. People find fine categorizations of information unwieldy and seemingly should be better suited to questions that have fewer answers than the ideal number of states for storage devices. But the casual observation that people will perform better with black-and-white distinctions is misleading: people should not always be routed to the questions with the fewest answers, nor are they in reality. It is only machines – and rarities like Holmes – that due to their flawlessness can always exploit the efficiency of binary devices and questions. These conclusions fit with the psychological finding that people can hold in consciousness only a small number of distinctions. Since Miller (1956), the estimate by psychologists of this number has been shrinking steadily, from 7 to roughly 4. See Herbert Simon (1974), Kahneman et al. (1992), Cowan (2000), and Treisman (2006). But this literature does not go so far as to claim that people are well-suited only to binary distinctions. We have evolved to carry a only few options in our minds, though not always just two, and that is in fact the efficient way to function.

In addition to organizations that process information à la Radner, our model of dynamic information elicitation applies to conventional decision-making: decision trees are formally interchangeable with trees of questions or trees that gather information. The upshot of this paper is that agents at each step should face only a few options and three options will often be best. This extension underscores a similarity to Gigerenzer (e.g., 1999) on frugal heuristics. Gigerenzer argues that the apparent limitations of human information processing, such as the bounds on what agents can hold in consciousness,

are decision-making advantages. The present paper makes this conclusion explicit: the coarse trees that people deploy in decision-making efficiently minimize costs.

Finally, efficient information elicitation is related to the optimality of languages. On the latter, the trade-off between the cost of communication and precision is studied in Crémer et al. (2007), Sobel (2015), and Dilmé (2017). Agents in the present paper in contrast must transmit information precisely and they use trees rather than words.

2 The static organization of information

The increasing returns advantage of coarse information sorting takes its simplest form in information storage and suggests why the information storage devices currently in use (transistors) attain two states. The head start of binary devices might now be difficult to overturn but the number of states in storage devices was once an engineering decision that could have been made differently. Theorem 1 below identifies a bias in favor of binary devices that can explain their ascendancy; we do not need to invoke an accident of technological history.

Suppose an array of N devices is available to store one out of $n > 1$ possible facts. If each device can assume k states it will partition the set of facts into k cells and the array as a whole can attain k^N configurations. Storage capacity thus shows increasing returns in the number of devices N. To ensure that each fact can be assigned a distinct configuration, we must have $k^N \geq n$. Let the cost of a device with k states equal $\kappa(k)$, which we assume satisfies $\kappa(k) > 0$ for $k > 1$. Storage costs will be minimized when the following problem is solved:

$$\min \kappa(k)N \quad \text{s.t.} \quad k^N \geq n$$

where k and N must be positive integers. To satisfy the $k^N \geq n$ constraint, k must be greater than 1 and the number of k-state devices should therefore equal the least integer N such that $k^N \geq n$, that is, $\lceil \log_k n \rceil$. So we could instead solve

$$\min_{k>1} \kappa(k) \lceil \log_k n \rceil.$$

Since we can associate the n facts with the numbers $0, ..., n-1$ and $\lceil \log_k n \rceil$ is the number of digits needed to write $n-1$ in base k, the natural engineering interpretation of this

problem is that it finds the optimal base to store one of n numbers. In this view, each device that can assume k states stores one k-ary or base-k digit.

Changes in k affect the two factors of $\kappa(k)\lceil \log_k n \rceil$ differently: the cost $\kappa(k)$ is presumably increasing in k while the number of devices $\lceil \log_k n \rceil$ is decreasing in k. How this trade-off is decided depends on the cost function. The optimality of a small value for k might seem to depend on a marginal cost of an additional state – raising k to $k+1$ – that increases in k, which is unlikely given economies of scale in production. But exponential increasing returns introduces a countervailing factor: the large-scale operation of devices with a small k might outweigh the gains due to diminishing marginal cost that can be reaped by large k devices. Formally, marginal costs are *increasing* if for all integers $k' > k > 0$,

$$\kappa(k'+1) - \kappa(k') \geq \kappa(k+1) - \kappa(k),$$

and are *decreasing* or *constant* if $\geq$ is replaced by $\leq$ or $=$ respectively. If the domain of κ were instead the reals greater than 1 then marginal costs will be increasing when κ is convex and decreasing when κ is concave. Both increasing and decreasing returns allow $\kappa(1) = 0$: a device that attains one state, which never varies and cannot store information, should be free.[3]

The force of increasing returns turns out to be sufficiently powerful that small k devices run at a large scale will prevail over the competing k's. In fact, under weak assumptions efficiency is achieved when devices attain two states. The marginal cost of additional states could diminish to 0 – increments to the base could become asymptotically free – and still it will be optimal to store information in bits.

Given a number of possible facts n, define (k, N) to be **feasible for** n if k and N are positive integers such that $k^N \geq n$. Since the storage cost of (k, N) is $\kappa(k)N$, define **k-state devices to store one out of n facts efficiently** if, for some N, (k, N) is feasible for n and $\kappa(k)N \leq \kappa(k')N'$ for all (k', N') that are feasible for n.

Theorem 1 *The following two statements are equivalent:*

- *for all integers $n > 1$, 2-state devices store one out of n facts efficiently,*

[3] A 1-state device in fact does not differ from a 0-state device.

- $\kappa(k) \geq \kappa(2) \lceil \log_2 k \rceil$ *for all integers* $k > 1$.

Storage in bits is therefore efficient if the cost of k-state devices rises quicker than $\log k$. Since $\log k$ is a concave function of k, a function κ with diminishing marginal costs can satisfy the log cost condition in Theorem 1; marginal costs can even converge to 0. That the efficiency of binary devices requires only their logarithmic superiority over their k-ary competitors has likely been instrumental to their predominance.

The argument for why the log cost condition, the second statement in Theorem 1, implies the first statement rests on increasing returns. We can use N devices with 2-states each to replace a k-state device if $2^N \geq k$ or equivalently if $N \geq \log_2 k$. The log cost condition simply states that, for the smallest such N, the 2-state devices are cheaper.

Proof of Theorem 1. Assume that $\kappa(k) \geq \kappa(2) \lceil \log_2 k \rceil$ for integers $k > 1$ and fix some positive integer n. Since $2^{\lceil \log_2 n \rceil} \geq 2^{\log_2 n} = n$, $(2, \lceil \log_2 n \rceil)$ is feasible for n.

Let (k', N') be feasible for $n > 1$. Since $2^{\lceil \log_2 k' \rceil} \geq 2^{\log_2 k'} = k'$, $\left(2^{\lceil \log_2 k' \rceil}\right)^{N'} \geq (k')^{N'}$ and hence $2^{\lceil \log_2 k' \rceil N'} \geq (k')^{N'}$. Since $(k')^{N'} \geq n$ by feasibility, $2^{\lceil \log_2 k' \rceil N'} \geq n$ and hence $\lceil \log_2 k' \rceil N' \geq \log_2 n$. Given that $\lceil \log_2 k' \rceil N'$ is an integer, $\lceil \log_2 k' \rceil N' \geq \lceil \log_2 n \rceil$. Since by assumption $\kappa(k') \geq \kappa(2) \lceil \log_2 k' \rceil$,

$$\kappa(k')N' \geq \kappa(2) \lceil \log_2 k' \rceil N' \geq \kappa(2) \lceil \log_2 n \rceil.$$

Thus 2-state devices store one out of n facts efficiently.

Conversely, assume for any integer $n > 1$ that 2-state devices store one out of n facts efficiently and suppose, for some $k' > 1$, that $\kappa(k') < \kappa(2) \lceil \log_2 k' \rceil$. Set $n = k'$ and observe that $(k', 1)$ and $(2, \lceil \log_2 n \rceil)$ are both feasible for n. Since $\kappa(k')(1) < \kappa(2) \lceil \log_2 n \rceil$, 2-state devices would not store one out of n facts efficiently. ■

Radix economy. Engineers in the 1950's studied the optimal number of states in information storage devices. The resulting literature on radix [base] economy assumed that the cost of storing a number that ranges from 0 to $n-1$ using base-k devices or digits equals $k \log_k n$: the cost of a k-ary device was taken to be k and the number of k-ary devices needed was approximated by $\log_k n$. The original work of Engineering Research Associates (1950) offered rationales for the implicit cost function $\kappa(k) = k$ but the functional form became canonical and was applied to any radix choice in computer architecture (see, e.g., Hurst (1984)). Although I will argue that this cost function is inappropriate for

information storage, the radix choice that it leads to will be indispensable for our main topic, the sequential organization of information.

The engineering literature was casual about the requirement that the base k and the number of devices N must be integers and therefore considered the problem:

$$\min_{k,N\in\mathbb{R}_+} kN \text{ s.t. } k^N = n \text{ or } \min_{k\in\mathbb{R}_+} k\log_k n.$$

It is easy to show that the solution to this problem occurs where k equals $e \approx 2.7$. Researchers then argued with extensive examples that the optimal integer value for k is usually 3. We can formalize this point by showing that, for all n sufficiently large, 3 is the optimal base k at a solution to the following *radix economy problem*

$$\min kN \text{ s.t. } k^N \geq n \text{ and } k \text{ and } N \text{ are positive integers.}$$

At any solution (k, N) to this problem, N must equal the least integer such that $k^N \geq n$, that is, $N = \lceil \log_k n \rceil$. We can therefore consider an optimal k to be a solution to the entire problem.

Theorem 2 *For all integers n sufficiently large, 3 is the unique base that solves the radix economy problem.*

Omitted proofs are in the Appendix. Theorems 1 and 2 are consistent since the radix economy literature's de facto cost function $\kappa(k) = k$ violates the log cost condition at $k = 3$.

Theorem 2 presents an empirical puzzle: if three is the optimal base, why are storage devices always binary rather than ternary? The explanation offered in the radix economy literature, e.g., Hallworth and Heath (1961), Frieder (1972), Parhami and McKeown (2013), is that the dominance of binary devices has been a mistake: ternary devices would improve efficiency.

An alternative to blaming engineers for failing to optimize is to challenge the cost function $\kappa(k) = k$. Even granting linearity in k, the function is difficult to justify. Since one of the states of a storage device will obtain by default, a k-state device has only $k-1$ active states and only these states should incur a cost. A $k = 1$ device for example does nothing, cannot store any number, and leaves the required number of devices undefined. Yet it has positive cost if $\kappa(k) = k$. If we try to repair the cost function by setting $\kappa(1) = 0$ while retaining $\kappa(k) = k$ for $k > 1$ then marginal costs become skewed: the

marginal cost of a second state will be 2 while the marginal cost of a third state will be 1. It is this implicit assumption that the marginal cost of active states drops markedly at $k = 3$ that drives the ternary optimality conclusion of Theorem 2.

For a cost function to meet the radix economy literature's default assumption that marginal costs are constant, which may be the neutral hypothesis from the engineering point of view, and to satisfy $\kappa(1) = 0$, we must set $\kappa(k) = k - 1$ for $k > 0$. I will call

$$\min_{k,N} (k-1)\,N \quad \text{s.t.} \quad k^N \geq n \text{ and } k \text{ and } N \text{ are positive integers}$$

the *modified radix economy problem*. It is easy to check that $\kappa(k) = k - 1$ satisfies the log cost condition of Theorem 1 and therefore $k = 2$ always solves the modified problem, again with N set to $\lceil \log_2 n \rceil$. In fact, $k = 2$ is normally the only solution.

Theorem 3 *For any integer $n > 1$, base 2 solves the modified radix economy problem and is the only solution except when n equals 3 or 9.*

Theorem 3 removes the bias in favor of 3-state devices: they are not cheaper than 2-state devices under a credible interpretation of constant marginal costs. Taken together, Theorem's 2 and 3 show that if the cost function shifts up from $k - 1$ to k the optimal k nevertheless increases.

The implausibility of large-state storage. There is no law of nature that dictates a $k - 1$ cost function or the log cost condition of Theorem 1: the technological facts of life might be such that 2-state devices are suboptimal. Suppose the log cost condition is violated, that is, for some $k \neq 2$,

$$\kappa(k) < \kappa(2)\,\lceil \log_2 k \rceil.$$

There will then exist integers n such that one out of n facts will be stored more cheaply with k-state devices than with 2-state devices (for example when $n = k$). Given that $\log_2 k$ increases so slowly, the small values of k provide the natural candidates to satisfy the above inequality. Since 3 is the smallest contender, the radix economy literature's advocacy of ternary computing has a partial defense: should binary storage ever be rejected, ternary storage is the likeliest replacement. As it happens, ternary computers were built or emulated from the late 1950's to the early 1970's, motivated in part by the fact that $k = 3$ solves the (original) radix economy problem (Hallworth and Heath (1961)); and

champions of ternary computing have argued they were cost effective.[4]

Even if ternary computers represent a lost opportunity, it is unlikely that k-state devices could store information efficiently when k is large. The presence of exponential increasing returns implies that the number of binary devices needed to replace a k-state device increases ever more slowly as k increases (since the derivative of $\log_2 k$ is proportional to $\frac{1}{k}$). Scenarios where $\kappa(k)$ is smaller than $\kappa(2) \lceil \log_2 k \rceil$ for large values of k are therefore highly implausible. Theorem 1 moreover applies to particular values of k rather than all $k > 1$. If $\kappa(\widehat{k}) \geq \kappa(2) \left\lceil \log_2 \widehat{k} \right\rceil$ for specific $\widehat{k}$'s (presumably the large values) then $\widehat{k}$-state devices can be efficiently replaced by 2-state devices.

Our treatment of information storage has supposed that each device can assume the same number of states. Mixed-radix arrays where devices can attain different numbers of states will be discussed in section 6.

3 The sequential organization of information

People organize information sequentially when they sift and categorize data or answer questions over time. To model the costs of processing information, I will follow the Radner (1993) assumption that the time spent on k items of information is a linear function of k, comparably to the engineering assumption that the cost of a storage device is proportional to the number of its states. That assumption was ill-matched to the storage setting: a computer in effect knows that if a k-state storage device has not assumed any of its $k-1$ nondefault states then it must assume its default state. Boundedly rational human agents, on the other hand, do not always draw such conclusions. When an agent can rule out $k-1$ out of k possible answers to a question, he or she cannot always infer what the remaining answer is or that it must be correct. I therefore assume to begin that the cost in time and cognition that an agent incurs when processing a question with k answers is proportional to k.

With this assumption in place, the problem of eliciting information from an agent at minimum cost resembles the radix economy problem. Although it would seem that

[4] Frieder (1972), Klimenko (1999), Parhami and McKeown (2013).

if bounded rationality raises the cost of answering a k-answer question from $k-1$ to k then the optimal k should fall in response, the classical and modified radix economy problems led to the opposite conclusion. Accordingly we will see that while machines due to their perfection can fully exploit the efficiency advantages of binary categorizations people cannot.

To make the information elicitation problem and our assumptions tangible, I consider an agent who consults a website or doctor for medical advice. The agent could describe every fact he knows about his health but that would be time-consuming and most of these facts are immaterial. So instead the website asks the agent a tree of questions. The first question might be 'is your problem physical, psychological, or both?'. If the answer is 'physical' the next question might ask 'do you suffer from fatigue, a respiratory problem, or some other condition?', if 'psychological' the next question might be 'are you experiencing depression, anxiety, or some other issue?'. And so forth. Each answer either brings the questions to a close or triggers a further question.

The same formal model arises in a variety of settings. In a decision tree rather a question tree, an agent would make a series of partial decisions among subsets of actions that progressively narrow down the agent's options. When choosing a vacation, for example, an agent could first choose among the options of a seaside, mountain, or urban holiday, and then, having picked the urban option, could choose between nearby and remote cities, and so on. Where a question tree pinpoints an agent's information, a decision tree pinpoints an action. As discussed in the introduction, the Radner (1993) bottom-up model of information processing in organizations would fit the present model if processing were serial rather than parallel (which can be a minor restriction as in Bolton and Dawatripont (1994)). A final variation, the routing of phone inquiries, will appear in section 5.

I assume the agent knows the true state ω in a state space Ω which, in the medical illustration, specifies the possible somatic facts the agent could have observed. To diagnose the agent, the website must determine which cell of a finite 'diagnosis partition' $\mathcal{P}_D$ of Ω contains ω. Each cell of $\mathcal{P}_D$ groups together all of the facts that the website deems to be unimportantly different, e.g., if eye color is irrelevant then each cell would contain

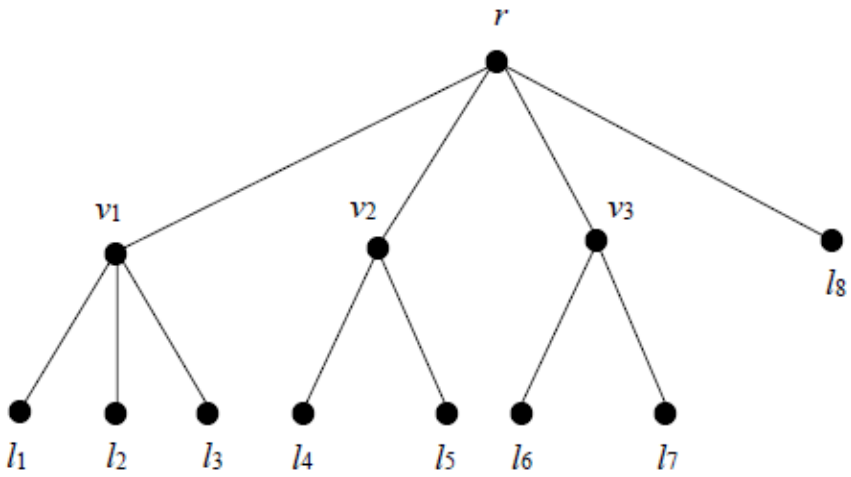

Figure 1: a question tree with root r, nonleaf nodes v_i, and leaves l_i.

all states that differ only with respect to the agent's eye color.

At each node v in a tree of questions T, the agent is either asked a question with answers that are represented by v's immediate successors or v is a leaf (terminal node). See Figure 1. Each node in T is also associated with an event in Ω: the root r is associated with Ω and the successors of each node v are associated with a partition of the event associated with v. The event that obtains at a node is interpreted as the set of the states consistent with the sequence of answers that leads to that node. So, for example, the 'physical' answer to question 1 followed by the 'fatigue' answer to question 2 identify both the sequence consisting of r and one of its immediate successors and the event in Ω where the agent has a physical problem that involves fatigue.

The tree T terminates in a *set of leaves* $L(T)$ and the family of events associated with $L(T)$ defines an 'answer partition' $\mathcal{P}_T$ of Ω. Since we can identify a leaf l of T with the sequence of nodes in T that form the unique *path* $p(l)$ from the root r to l, the cells of $\mathcal{P}_T$ lie in a one-to-one correspondence with the paths of T or, equivalently, with the sequences of answers that the agent can supply.[5]

For a tree to diagnose correctly, the answer partition $\mathcal{P}_T$ must refine the diagnosis partition $\mathcal{P}_D$: each cell of $\mathcal{P}_T$ should contain only states in one cell of $\mathcal{P}_D$. The agent's answers will then reveal which cell of $\mathcal{P}_D$ contains the true state. Letting $\mathcal{P}_D$ have n

[5]While $p(l)$ depends on the tree T, there will be no ambiguity about what tree is under consideration. A path in this paper will always indicate a sequence of nodes from root to leaf rather than an arbitrary sequence of linked nodes.

cells, achieving this goal will require a tree T to have at least n leaves. Since having more than n leaves can only raise costs, there is no disadvantage in letting T have exactly n leaves and setting the answer partition $\mathcal{P}_T$ to equal $\mathcal{P}_D$ itself. I will call a tree T with n leaves *revealing*.

Our initial assumption will be that the time it takes to answer a question is proportional to the number of answers an agent can give and hence has to read. Agents thus do not deduce the meaning of the final answer of a question with k answers from the meaning of the first $k-1$ answers. Although the k possible answers form a partition of the event the agent has identified via his earlier answers, the exact boundaries of the cells of this partition will be conveyed only by the entire list of answers. For example, if a tree of questions has determined at some node that an agent suffers from tinnitus and asks 'is the ringing in your ears (a) loud, (b) soft, or (c) barely audible?' the agent would need to read option (c) in order to understand exactly what is meant by options (a) and (b). Even when agents know their symptoms with precision, they still need to learn the words and syntax that questions use to describe symptoms, and that requires seeing which informally described events the tree deems to be distinct. In addition, what constitutes an exhaustive list of states can vary from person to person and organization to organization. By reading all the answers to a question, agents can discover which states the website fuses and which it distinguishes among, e.g., whether there is more than one type of lower back pain.[6]

Given the proportional cost assumption, if the true state lies in the event associated with leaf l of the tree T and k_v is the number of immediate successors of a node v in T (the number of answers the agent can give to the question asked at v) the total amount of time the agent will spend answering questions will be proportional to $\sum_{v \in p(l)} k_v$.

[6] For the cost of a question with k answers to be proportional to k, there must also be concise language that can express k answers. Human languages are supple, however, particularly when describing coarse sets of options. If an agent suffers from pain and the questioner wants to determine its exact location, everyday language provides several dividing lines that can partition a body, e.g., neck, waist, knees, feet. An excessively fine partition of locations however would require cumbersome language. The proportionality of costs to k therefore becomes less tenable as numbers of answers increase. But the nonlinearities I abstract away from will further favor the questions with small numbers of answers that I will show to be optimal. Suppose the goal is to determine the location of pain in the arm down to the centimeter. Rather than asking an agent one question that specifies 100 locations, it will be far quicker to determine first the approximate location of the pain – hand, wrist, forearm, elbow, upper arm – before asking for more precision.

4 Minimax optimality

The information elicitation problem seeks a revealing tree of questions that takes the shortest amount of time for an agent to go through. Following a longstanding computer science tradition, we will begin by letting the cost of an algorithm equal the greatest amount of time it can take to proceed through the tree; in the Radner bottom-up interpretation of a tree, where there is no formal uncertainty, this will be the time it takes for an organization to complete its information processing. Fixing a diagnosis partition $\mathcal{P}_D$ with n cells, a *minimax optimal tree* is a tree T that minimizes the cost

$$\kappa_M(T) = \max_{l \in L(T)} \sum\nolimits_{v \in p(l)} k_v$$

subject to the constraint that T is revealing (has n leaves).

The simplest way to find a minimax optimal tree is via a minimization problem closely related to the classical radix economy problem. Each leaf l of a tree T defines a *path of (numbers of) answers* $k_1, ..., k_N$ given by the k_v that occur as v proceeds along the path from the root to l. Define a path of answers $k_1, ..., k_N$ to be a *maximum-cost path* of T if $\sum_{i=1}^{N} k_i = \kappa_M(T)$ and $k_1 \times \cdots \times k_N \geq n$.

The maximum-cost paths of a minimax optimal tree are found by solving

$$\min \sum_{i=1}^{N} k_i \text{ s.t. } k_1 \times \cdots \times k_N \geq n, \tag{MIN}$$

where N and each k_i must be positive integers. This problem displays the form of increasing returns characteristic of the sequential organization of information: the number of diagnoses that can be achieved by questions with $k_1, ..., k_N$ answers is $k_1 \times \cdots \times k_N$, an exponential function of the geometric mean of the k_i.

Suppose $(k_1^*, ..., k_{N^*}^*)$ solves MIN and let T^* be a revealing tree such that every path of answers begins with $k_1^*, ..., k_{N^*-1}^*$ and ends with a $k_{N^*} \leq k_{N^*}^*$.[7] Since the final component k_{N^*} must equal $k_{N^*}^*$ on some paths, $\kappa_M(T^*) = \sum_{i=1}^{N^*} k_i^*$. The proof of the Lemma below

[7] If $\prod_{i=1}^{N^*} k_i^* > n$ then to ensure that T^* has exactly n leaves k_{N^*} must fall short of $k_{N^*}^*$ on some paths. We could alternatively let revealing trees have more than n leaves and require only that $\mathcal{P}_T$ refines $\mathcal{P}_D$. To ensure that there exist $\mathcal{P}_T$ with $\prod_{i=1}^{N^*} k_i^*$ leaves, we would then need to assume that $|\Omega| \geq \prod_{i=1}^{N^*} k_i^*$.

shows that every revealing tree T must have *some* path of answers $k'_1, ..., k'_N$ that satisfies $k'_1 \times \cdots \times k'_N \geq n$ and so $\kappa_M(T) \geq \sum_{i=1}^{N} k'_i$. Consequently T^* is minimax optimal: T^* minimizes κ_M among revealing trees. Conversely, if T is minimax optimal then some path of answers $k_1, ..., k_N$ of T satisfies $k_1 \times \cdots \times k_N \geq n$ and any such $k_1, ..., k_N$ must solve MIN: if not then $\sum_{i=1}^{N} k_i$ and hence $\kappa_M(T)$ would be greater than $\kappa_M(T^*) = \sum_{i=1}^{N^*} k_i^*$ for the T^* identified above.

Lemma 1 *For any n, there exists a minimax optimal tree with a maximum-cost path of answers $k_1, ..., k_N$ if and only if $k_1, ..., k_N$ solves MIN.*[8]

The Lemma shows that finding a minimax optimal tree is similar to solving the classical radix economy problem (where the cost of a k-state device equals k): the only difference is that in the radix economy case every device has to assume the same number of states k while here each question can have a different number of answers. Had we allowed each device that stores information to assume a device-specific number of states then the classical radix economy problem would be exactly the MIN problem. We will consider why mixed-radix arrays do not arise in practice in section 6.

The increasing returns implicit in the constraint of problem MIN forces the solution values of k_i to be small: no k_i can be greater than 5. Consequently no question on a maximum-cost path of an optimal tree can have more than 5 answers: it is cheaper to scale up the usage of questions with fewer answers. For example, $k_j = 6$ cannot occur at a solution to MIN since $\sum_{i=1}^{N} k_i$ will fall if 6 is replaced by the pair $(k_j = 3, k_{j'} = 2)$: the subtree that consists of a ternary (3-answer) question followed by three 2-answer questions has the same number of leaves as a 6-answer question but a cost of 5 rather than 6. The smaller k_i's lead to a cost reduction even when their product (the potential number of diagnoses) overshoots n. For example, when $n = 7$ it is cheaper to set $k_1 = k_2 = 3$ (a sum of 6) than to use one component $k_1 = 7$: two ternary questions in sequence dominate one 7-answer question even though the ternaries can generate 9 diagnoses.

Given the link between MIN and the classical radix economy problem, it comes as no

[8] We can also interpret MIN as the problem of minimizing cost where, for each i, the ith question on any path must have the same number of answers k_i. With this interpretation, if $k_1 \times \cdots \times k_N > n$ at a solution to MIN then the number of answers, say to the final question, would need to be pruned on some paths.

surprise that ternary questions dominate the maximum cost paths of minimax optimal trees when n is large. Theorem 2 bounded the number of k-state devices with $k \neq 3$ that can be used to store information efficiently. We can use this bound to conclude that on a maximum-cost path of an optimal tree the number of nonternary questions – nodes v with $k_v \neq 3$ – must also be bounded, independently of n. According to Theorem 2 there is a $\overline{n}$ such that for $n \geq \overline{n}$ the only solution to the classical radix economy problem occurs at $k = 3$. The number of k_i's that can equal a particular $k \neq 3$ at a solution of MIN can therefore be no greater than $\lceil \log_k \overline{n} \rceil$: if more were used then we could lower $\sum_i k_i$ without decreasing $\prod_i k_i$ by replacing the k's with sufficiently many k_i's equal to 3. Since each k_i must also be less than 6, the bound on the number of k_i's that can equal a particular $k \neq 3$ leads to a bound on the total number of k_i's not equal to 3.

The following Theorem sums up the productivity of increasing returns and the advantage of ternary questions in particular.

Theorem 4 *The number of nonternary questions on any maximum-cost path p of any minimax optimal tree is bounded independently of n and no question on p has more than* 5 *answers.*

Minimax optimal trees can nevertheless display considerable diversity. The number of answers to questions that lie off the maximum-cost paths can increase without bound as n increases. And although maximum-cost paths must eventually be dominated by 3's as n increases, that preponderance does not occur for modestly sized n's. Suppose the number of leaves n lies between 2 and 100 and that, except for the final question, the ith question on any path always has the same number of answers. The final question is allowed to have different numbers of answers on different paths to ensure that trees have exactly n leaves. We can then count the number of i's such that the ith question asked is ternary (declaring the final question to be ternary when a majority of the final question nodes are ternary). It turns out that when n equals any of 55 integers in $[2, 100]$ there is a minimax optimal tree with at least as many nonternary questions as ternary questions.[9]

[9] The 55 integers are 2, 4-8, 10-16, 19-20, 28-40, 55-72, and 82-90. The assumption that each question but the last has a common number of answers ensures that the test does not exploit the potential of minimax optimal trees to have idiosyncratic numbers of answers on the paths with a cost that falls short of the minimax cost.

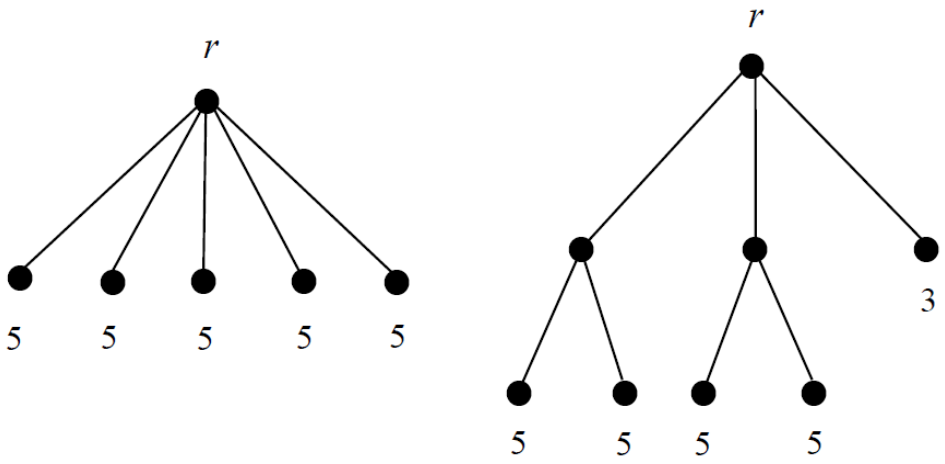

Figure 2: two five-leaf trees with the costs of each path.

5 Expectation optimality

While the minimax definition of cost makes sense in the Radner bottom-up interpretation of information processing, in the information elicitation interpretation it suffers from a standard drawback of objective functions determined by worst-case outcomes. An agent can spend a needlessly large amount of time answering some paths of questions without affecting $\kappa_M(T)$, the amount of time it takes to answer the most time-consuming path. To illustrate with $n = 5$, the tree formed by one 5-answer question and the tree formed by one 3-answer question followed by two 2-answer questions both have maximum-cost paths with a cost of 5 but the latter tree enjoys one cheaper path with cost 3. See Figure 2 where each leaf l lists the cost of the path that leads to l. Another curious consequence of the minimax objective is that as n increases there is no upper bound on the number of 2-answer questions on some paths.

We therefore turn to an alternative measure of the cost of answering a tree of questions T, the sum across paths of the number of answers the agent can give or, equivalently, the expectation of the number of answers an agent encounters if each leaf is assigned equal probability weight,

$$\kappa_E(T) = \sum\nolimits_{l \in L(T)} \sum\nolimits_{v \in p(l)} k_v.$$

Multiplying by $\frac{1}{n}$ leads to the expectation interpretation. The number $\kappa_E(T)$ is known as 'degree path length' in tree theory (Knuth (1998b)). Again fixing n, a tree T that

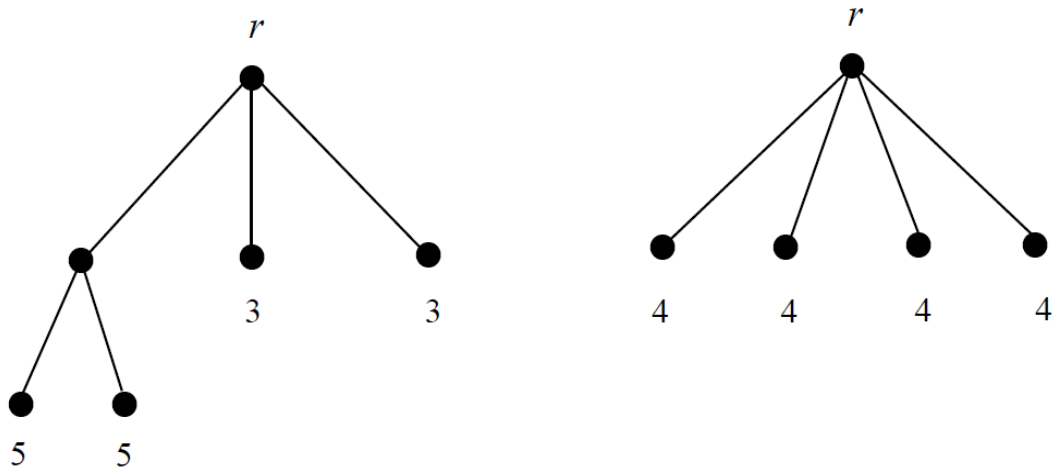

Figure 3: an expectation-optimal and an expectation-and-maximin-optimal tree.

minimizes κ_E subject to being revealing is *expectation optimal*. Due to the right-hand tree of Figure 2, the suspicious left-hand tree can never form part of an expectation-optimal tree. While expectation optimality does not imply minimax optimality – compare the expectation-optimal tree on the left of Figure 3 with the tree on the right that is both expectation and minimax optimal – it comes close.

Proposition 1 *For any n, expectation-optimal tree T, and minimax-optimal tree $\widehat{T}$, $\kappa_M(T)$ is no more than 1 greater than $\kappa_M(\widehat{T})$.*

Expectation optimality sharply reduces the scope for nonternary questions. As Figure 2 indicates, questions with 5 answers can no longer occur in an optimal tree. And while it cannot be expectation optimal to ask only ternary questions unless n is an exact multiple of 3, a classical result in tree theory, due to Knuth (1998b), reports that there exists an expectation optimal tree that is nearly all-ternary: on each path there is at most one nonternary exception, a binary question. For our purposes however what matters is whether it could also be optimal to use several nonternary questions. The answer is effectively 'no': each path of an optimal tree can ask no more than 2 nonternary questions, which amounts to a converse of Knuth's theorem.

Theorem 5 *For any n and any expectation optimal tree T, each path of T can include at most two nonternary questions.*

To see an example of this surprising conclusion, suppose that $n = 8$. The tree defined by three binary questions in a row, which has a cost per leaf of 6, would then seem to be ideal since $2^3 = 8$. But the 9-leaf tree defined by two ternary questions in a row also has a per-leaf cost of 6 and, by pruning a leaf from the latter tree (converting one of its final ternary questions into a binary question), we can reduce the per-leaf cost to $5\frac{3}{4}$. So the all-binary tree is indeed suboptimal.

An illustration. The trees of options that route the callers to a business or organization, which I will call 'phone trees,' are formally interchangeable with trees of questions that elicit information. If a caller wants to reach a particular department or hear a specific recorded message and the business wants to connect him or her as quickly as possible, what sort of phone tree should be used? If callers must listen to every option at every node on the path they follow then the time taken by a path – its cost – will be proportional to $\sum_{v \in p(l)} k_v$, where the leaf l now indicates the message or department the caller seeks. The connection between phone trees and information storage or elicitation was made by Hayes (2001) though in his model, which was designed to be normative, every node presents callers the same number of options, a uniformity that phone trees rarely if ever display.

In a sample of 10 UK phone trees, I found that 3 were minimax optimal of which one was also expectation optimal and 7 were suboptimal. The greatest number of options at any node across the sample was 6 (which occurred once), one more than the greatest number that can occur on the maximum cost paths of minimax optimal trees. And the number of binary options on a path never exceeded one: phone trees do not display the dominant binariness of optimal arrays of mechanical storage devices. The constraints of language likely block the exact predictions of minimax or expectation optimality. If the words needed to regroup $k > 3$ sets of messages into 3 sets save no time (because there is no concise way to describe the latter sets) it will not improve efficiency to replace a k-ary with a ternary partition of messages.

6 When binary questions are efficient

The more-than-binary predictions of the minimax and expectation definitions of optimality accord with how information is actually elicited, and the prediction of binary storage accord with the facts of mechanical storage devices. Eliciting information from people frequently uses more-than-binary questions, while information storage in computers always employs bits.

There are nevertheless prominent cases, even in the terrain of medicine, where agents are asked binary questions. A common feature of a visit to a doctor's or dentist's office is the long list of yes-or-no questions about preexisting conditions, drug allergies, and current drug prescriptions. Since preexisting conditions usually refer to the explicit diagnoses of other physicians, the 'yes' answers to all of these questions are nearly unambiguous. Consequently the 'no' answers do not need to be spelled out as well. So, for example, 'are you allergic to penicillin?' can be answered with yes and no checkboxes rather than with separate descriptions of the two answers. Faced with such questions, agents perform as well as Sherlock Holmes: they can deduce the meaning of the 'no' answer from the content of the 'yes' answer without error. The cost of unambiguous binary questions should therefore equal the time it takes to read one answer, not two. Given the results of section 2, a Holmesian specification of costs suggests that optimal questions will be binary: questions with the minimum numbers of answers can harness the full effectiveness of increasing returns. If this result is correct, we would then have another case where lowering a cost function from k to $k-1$ lowers rather than raises the optimal k.

To test these conjectures, we briefly lay out the optimization results that hold when the cost of a k-answer question is proportional to $k-1$ rather than k. Even straightforward queries about a patient's medical history *could* have nonbinary answers: replies could subdivide preexisting conditions into a fine list of categories. The optimization problem therefore needs to be solved explicitly, and it is not covered by section 2 since the questions agents face arrive sequentially.

A $k-1$ cost assumption can apply to Radner (1993) as well. While it may take k units of time to read k items of information, the processing of k items of information may

well take $k-1$ time units. In the associative operations, such as addition or finding a maximum, that Radner had in mind, the processing of a single item – where no operation in fact takes place – should presumably be free.

The consequences of costs proportional to $k-1$ are that any expectation optimal tree must be all-binary – every node is a binary question – and the maximum cost paths of minimax optimal trees must be nearly all-binary. The diversity displayed by minimax optimal trees when costs are proportional to k thus disappears. Lists of binary medical questions in fact provide remarkably efficient elicitation tools. A list of 30 questions can be in answered in 2^{30} ways – more than a billion patient types can be distinguished – but most individuals can proceed through such lists in seconds.

Define a *minimax H-optimal tree* to be a T that minimizes the cost

$$\kappa_M^*(T) = \max_{l \in L(T)} \sum\nolimits_{v \in p(l)} (k_v - 1)$$

subject to the constraint that T is revealing. An *expectation H-optimal tree* is a T that minimizes

$$\kappa_E^*(T) = \sum\nolimits_{l \in L(T)} \sum\nolimits_{v \in p(l)} (k_v - 1)$$

again subject to the constraint that T is revealing.

As with minimax optimal trees, a minimax H-optimal tree can be found by solving a simple optimization problem

$$\min \sum\nolimits_{i=1}^{N} (k_i - 1) \text{ s.t. } k_1 \times \cdots \times k_N \geq n, \qquad \text{(MIN}^*\text{)}$$

where N and the k_i must be positive integers. If we redefine a path of answers $k_1, ..., k_N$ of a tree T to have *maximum cost* when $\sum_{i=1}^{N} (k_i - 1) = \kappa_M^*(T)$ and $\prod_{i=1}^{N} k_i \geq n$, the earlier Lemma and its proof extend: a minimax H-optimal tree can have a maximum-cost path $k_1^*, ..., k_{N^*}^*$ if and only if $k_1^*, ..., k_{N^*}^*$ solves MIN*.

Any solution $k_1^*, ..., k_{N^*}^*$ of MIN* must consist almost entirely of 2's. Due to Theorem 3, replacing any selection of candidate k_i^*'s that does not consist of 2's with 2's will, outside of a few exceptions, lead to a strict improvement. In the exceptions that arise

for some values of n, one or two k_i^*'s can equal 3 while the remainder must equal 2.[10] And when there is an exception, a solution of all 2's also exists. Maximum-cost paths of minimax H-optimal trees thus can be all-binary and must be nearly all-binary.

The expectation H-optimal trees are even more dominated by binary questions: costs will fall if *any* ternary question appearing at any node is replaced by binary questions (one binary node followed by a second binary node at one branch of the first binary node). And any expectation H-optimal tree is also minimax H-optimal. Define a tree T to have *nearly equal path lengths* if for any two paths of T the number of nodes on the paths differ by at most 1.

Theorem 6 *For any n, a tree T is expectation H-optimal if and only if T is revealing, all-binary, and has nearly equal path lengths. Any such T is also minimax H-optimal. Finally, on any maximum-cost path of a minimax H-optimal tree there can be at most two nonbinary questions and each must be ternary.*

The problem MIN* is the cost minimization problem that an efficient mixed-radix array of storage devices must solve, assuming that the $k_i - 1$ specification of cost holds. Theorem 6 thus explains why mixed-radix storage devices never arise in practice: they bring no advantage relative to all-binary arrays. See Knuth (1998a) on the possibility of mixed-radix computers.

7 Conclusion

To exploit the exponentially increasing returns that occur when information is sorted a large number of times, both machines and human agents should categorize information coarsely. Machines can go one step further by taking advantage of the fact that a default option on a storage device comes for free: they can therefore enjoy the full benefit of the increasing returns that accompany binary sorting. People are more complicated. In settings where binary questions have a meaning that can be absorbed without ambiguity after a description of only one answer, people can achieve the efficiency of binary catego-

[10] Theorem 3 per se does not rule out the possibility that some k_i^* could equal 9. But a solution with $k_i^* = 9$ cannot occur since it is dominated by a replacement consisting of four 2's.

rizations. But in most cases, people have to learn the meaning of options by reading and absorbing each and every answer they can give to a question. The extra time imposes a cost and the form of efficient questions will change as a result. Although it would seem that this increase in the cost of answers would pare down the efficient number of answers, the optimum moves the other way. The number of replies in efficient questions increases when the cost of a question with k answers rises from $k-1$ to k.

A Appendix: remaining proofs

Proof of Theorem 2. Define $g : \mathbb{R}_{++} \to \mathbb{R}$ by $g(l) = 3(\log_3 l + 1)$. For each $k \in \mathbb{Z}_{++}$, define $f_k : \mathbb{R}_{++} \to \mathbb{R}$ by $f_k(l) = k \log_k l$. Let $(\widehat{k}, \widehat{N})$ be a solution to the radix economy problem for some n.

Suppose first that $\widehat{k} > 6$. Since $g(9) = 9$ and $g'(l) < 1$ for all $l \geq 9$, $g(l) < l$ for any $l > 9$. So if $\widehat{k} > 9$ then $3\left\lceil \log_3 \widehat{k} \right\rceil < \widehat{k}$. For $\widehat{k} \in \{7, 8, 9\}$, $3\left\lceil \log_3 \widehat{k} \right\rceil = 6 < \widehat{k}$. So for each $\widehat{k} > 6$, $3\left\lceil \log_3 \widehat{k} \right\rceil \widehat{N} < \widehat{k}\widehat{N}$. Since in addition $3^{\left\lceil \log_3 \widehat{k} \right\rceil \widehat{N}} = \left(3^{\left\lceil \log_3 \widehat{k} \right\rceil}\right)^{\widehat{N}} \geq \left(\widehat{k}\right)^{\widehat{N}} \geq n$, $(\widehat{k}, \widehat{N})$ could not be a solution.

Next suppose $\widehat{k} \in \{2, 5, 6\}$. In each case, there exists a $\bar{l} \in \mathbb{R}_{++}$ such that $f_{\widehat{k}}(\bar{l}) = g(\bar{l})$ and, for all $l > 0$, $f'_{\widehat{k}}(l) > g'(l)$. Hence $\widehat{k}\left\lceil \log_{\widehat{k}} n \right\rceil > 3\lceil \log_3 n \rceil$ for $n > \bar{l}$ and therefore $\widehat{k}\widehat{N} > 3\lceil \log_3 n \rceil$ for $n > \bar{l}$. Since $3^{\lceil \log_3 n \rceil} \geq n$, $(\widehat{k}, \widehat{N})$ could not be a solution when $n > \bar{l}$. (The greatest n such that some $(\widehat{k}, \widehat{N})$ provides a solution is $n = 6^2$ when $\widehat{k} = 6$, $n = 5^3$ when $\widehat{k} = 5$, and $n = 2^{27}$ when $\widehat{k} = 2$.)

Finally if $\widehat{k} = 4$ then $(2, 2\widehat{N})$ would also be a solution (since $4^{\widehat{N}} = 2^{2\widehat{N}}$) which we have shown cannot occur for all n sufficiently large. ■

Proof of Theorem 3. Adapting the notation from the previous proof, define $g : \mathbb{R}_{++} \to \mathbb{R}$ by $g(l) = 1 + \log_2 l$ and $f_3 : \mathbb{R}_{++} \to \mathbb{R}$ by $f_3(l) = 2\log_3 l$. Let $(\widehat{k}, \widehat{N})$ be a solution for some n.

Suppose that $\widehat{k} > 3$. Since $g(4) = 3$ and $g'(l) < 1$ for all $l \geq 4$, $g(l) < l - 1$ for any $l > 4$. Hence, for $\widehat{k} > 4$, $\left\lceil \log_2 \widehat{k} \right\rceil < \widehat{k} - 1$. Since $\lceil \log_2 4 \rceil = 2 < 3$, we have $\left\lceil \log_2 \widehat{k} \right\rceil \widehat{N} < \left(\widehat{k} - 1\right)\widehat{N}$ for any $\widehat{k} > 3$. Since $2^{\left\lceil \log_2 \widehat{k} \right\rceil \widehat{N}} = \left(2^{\left\lceil \log_2 \widehat{k} \right\rceil}\right)^{\widehat{N}} \geq \left(\widehat{k}\right)^{\widehat{N}} \geq n$, $(\widehat{k}, \widehat{N})$ could not be a solution.

Now let $\widehat{k} = 3$. Then $f_3(15) > g(15)$ and, for all $l > 0$, $f'_3(l) > g'(l)$. Hence $2\lceil \log_3 n \rceil > \lceil \log_2 n \rceil$ for $n > 14$ and therefore $2\widehat{N} > \lceil \log_2 n \rceil$ for $n > 14$. Since $2^{\lceil \log_2 n \rceil} \geq n$, $(3, \widehat{N})$ could not be a solution when $n > 14$. For $n \in \{4, ..., 8, 10, ..., 14\}$, $2\lceil \log_3 n \rceil > \lceil \log_2 n \rceil$ and so again $(3, \widehat{N})$ could not be a solution. For $n \in \{3, 9\}$, $2\lceil \log_3 n \rceil = \lceil \log_2 n \rceil$ and so $(3, 1)$ is a solution when $n = 3$ and $(3, 2)$ is a solution when $n = 9$. ■

Proof of Lemma 1. We first show by induction that if $\mathcal{P}_D$ has n cells then a revealing

tree T must have a path where the product of the number of answers to the questions asked is at least n, that is, $\prod_{v\in p(l)} k_v \geq n$ for some leaf l of T. The number of leaves n_v that descend from a node v with only leaf descendants is no greater than k_v. So assume the following induction hypothesis: for node v' in T and each v_c that is a child of v' there exists a leaf $l(v_c)$ of the subtree $T(v_c)$ with root v_c such that the number of leaves n_{v_c} that descend from v_c is no greater than $\prod_{v\in p_{T(v_c)}(l(v_c))} k_v$, where $p_{T(v_c)}(l(v_c))$ denotes the path from v_c to $l(v_c)$. Then the number of leaves that descend from v' is no greater than $k_{v'} \times \max\{\prod_{v\in p_{T(v_c)}(l(v_c))} k_v : v_c \text{ is a child of } v\}$. Let v_c^* denote the argmax v_c and set $l(v') = l(v_c^*)$, which is a leaf of the subtree $T(v')$ with root v'. Then the number of leaves $n_{v'}$ that descend from v' is no greater than $\prod_{v\in p_{T(v')}(l(v'))} k_v$.

Suppose $(k_1^*, ..., k_{N^*}^*)$ solves MIN and define T^* as follows: for $1 \leq i < N^*$ let the ith question on any path have k_i answers and let the maximum number of answers to the N^*th question be $k_{N^*}^*$, with the number of answers to the N^*th question on some paths reduced from $k_{N^*}^*$ if necessary to ensure that T^* has exactly n leaves. To see that T^* is minimax optimal, suppose to the contrary that some revealing tree T' such that $\kappa_M(T') < \kappa_M(T^*)$. Given the result of the previous paragraph, there must be a leaf l of T' such that $\prod_{v\in p(l)} k_v \geq n$. Moreover $\kappa_M(T^*) = \sum_{i=1}^{N^*} k_i^*$ and $\kappa_M(T') \geq \sum_{v\in p(l)} k_v$ and therefore $\sum_{i=1}^{N^*} k_i^* > \sum_{v\in p(l)} k_v$, contradicting the assumption that $(k_1^*, ..., k_{N^*}^*)$ solves MIN.

Conversely suppose T is minimax optimal with a maximum-cost path $k_1, ..., k_N$. If there were a $(k_1', ..., k_{N'}')$ such that $\sum_{i=1}^{N'} k_i' < \sum_{i=1}^{N} k_i$ and $\prod_{i=1}^{N'} k_i' \geq n$ then define T' as follows. Let N'' be the minimum integer such that $\prod_{i=1}^{N''} k_i' \geq n$. Set, for $1 \leq i < N''$, the ith question on any path to have k_i' answers and let the answers to the N''th question be pruned so that T' has n leaves. Then $\kappa_M(T') < \kappa_M(T)$ and T' can be revealing. ■

Proof of Theorem 4. We first show that if $(k_1, ..., k_N)$ is a solution to MIN there can be no $k_j > 6$. If $k_j > 6$ then the proof of Theorem 2 shows that $3\lceil \log_3 k_j \rceil < k_j$. Consequently the objective $\sum_i k_i$ decreases if k_j is eliminated, the other k_i's remain unchanged, and we include $\lceil \log_3 k_j \rceil$ components equal to 3. Moreover, since $3^{\lceil \log_3 k_j \rceil} \geq k_j$, $3^{\lceil \log_2 k_j \rceil} \prod_{i\neq j} k_i \geq \prod_{i=1}^{N} k_i \geq n$. A $k_j = 6$ also cannot occur at a solution: $\sum_i k_i$ will decrease if k_j is eliminated, the other k_i's remain unchanged, and we include a 3 component and a 2 component, and $(3 \times 2) \prod_{i\neq j} k_i \geq \prod_{i=1}^{N} k_i \geq n$.

Theorem 2 establishes that there is a $\overline{n}$ such that for all $n \geq \overline{n}$ and all positive integers $k \neq 3$, $3\lceil \log_3 n \rceil < k \lceil \log_k n \rceil$. Define $\overline{N}_k = \lceil \log_k (\overline{n} - 1) \rceil$ and suppose at a solution $(k_1, ..., k_N)$ there are $N_k > \overline{N}_k$ components equal to the integer $k \neq 3$. Since $k^{N_k} > k^{\overline{N}_k} \geq k^{\log_k(\overline{n}-1)} = \overline{n} - 1$, or $k^{N_k} \geq \overline{n}$, we have $3\lceil \log_3 k^{N_k} \rceil < k \lceil \log_k k^{N_k} \rceil = kN_k$. Since in addition $3^{\lceil \log_3 k^{N_k} \rceil} \geq 3^{\log_3 k^{N_k}} \geq k^{N_k}$, replacing the N_k components equal to k with $\lceil \log_3 k^{N_k} \rceil$ components equal to 3 lowers the objective $\sum_i k_i$ without decreasing $\prod_i k_i$. Hence $\overline{N}_k$ bounds the number of components that can equal $k \neq 3$ at a solution to MIN and so the largest number of k_i's at a solution not equal 3 can be no greater than $\overline{N}_2 + \overline{N}_4 + \overline{N}_5$.

A $k_j = 5$ can be optimal. When $n = 5$, for example, the value of the objective function, 5, when there is a single 5 component is strictly less than the value when there are only 2 or only 3 components, $2\lceil \log_2 5 \rceil = 3\lceil \log_3 5 \rceil = 6$, or only 4 components,

$4\lceil \log_4 5 \rceil = 8$ and is tied with the value 5 when there is one 2 component and one 3 component. ■

Proof of Proposition 1. Fix n throughout. We first show that if T is expectation optimal then $\left|\sum_{v\in p(l)} k_v - \sum_{v\in p(l')} k_v\right| \leq 2$ for all leaves l and l' of T. If to the contrary there are leaves of T such that $\sum_{v\in p(l')} k_v = c$ and $\sum_{v\in p(l)} k_v \geq c+3$, then there is an improvement T' defined as follows: (i) replace l' with a binary node with two leaves as children, (ii) remove l, and (iii) if in T the parent v^* of l is binary, let the subtree with root v^* of T' be the subtree of T whose root is the sibling of l. The change in κ_E when T' replaces T due to (i) is $2(c+2) - c = c+4$. Let $l, v_1, ..., v_{k_{v^*}-1}$ be the children of v^*, and for $i = 1, ..., k_{v^*} - 1$ let T_i be the subtree of T with root v_i and let L_i be the number of leaves in T_i. The change in κ_E due to (ii) and (iii) is then

$$\kappa_M(T_1) + n_1(c+3-2) - (c+3+\kappa_M(T_1) + n_1(c+3)) = -(c+3) - 2n_1 \leq -(c+5)$$

when v^* is binary and

$$\sum_{i=1}^{k_{v^*}-1} (\kappa_M(T_i) + n_i(c+3-1)) - \left(c+3+\sum_{i=1}^{k_{v^*}-1} (\kappa_M(T_i) + n_i(c+3))\right) = -(c+3) - \sum_{i=1}^{k_{v^*}-1} n_i \leq -(c+5)$$

when v^* is not binary. In both cases T' is therefore an expectation improvement.

Letting m denote $\kappa_M(\widehat{T})$ where $\widehat{T}$ is minimax optimal, the proof of the Lemma shows that there is a tree T^* with a leaf l such that $\prod_{v\in p(l)} k_v \geq n$ and $\sum_{v\in p(l)} k_v \leq m$. Let the number of answers along $p(l)$ be given by $k_1, ..., k_{N^*}$. Define the revealing tree $\widehat{T}$ by letting the ith question for $1 \leq i < N^*$ on any path have k_i answers and letting the maximum number of answers to the N^*th question equal k_{N^*} (with the number of answers reduced from k_{N^*} on some paths so that $\widehat{T}$ has n leaves). Then $\kappa_E(\widehat{T}) \leq mn$. Now let T be expectation optimal and suppose $\kappa_M(T) \geq m+2$. Then by the previous paragraph $\sum_{v\in p(l)} k_v \geq m+1$ for every leaf l in T and hence $\kappa_E(T) \geq n(m+1)$, a contradiction. ■

Proof of Theorem 5. To state some known preliminaries about expectation optimal (henceforth 'optimal') trees, we begin with some definitions. It is helpful to represent a number of leaves n as $3^k + s$, where $k = \lfloor \log_3 n \rfloor$ and $0 \leq s < 2 \times 3^k$. A node v of a tree is *binary* if v has two successors and *ternary* if v has three successors. A tree with root r is *binary-rooted* if r is binary and *ternary-rooted* if r is ternary. Let $\kappa(n)$ denote the cost of an optimal tree with n nodes. Define the *marginal cost* $\kappa'(d)$ *of the integer* $d \geq 1$ to equal $\kappa(d+1) - \kappa(d)$ and define *marginal costs to be constant between integers* $a \geq 0$ *and* $b \geq 0$ where $b > a$ if there exists an integer c such that, for all integers $m \geq 0$, $\kappa'(m) = c$ if and only if $m \in [b-1, a]$.

The preliminaries are: (1) if $n \in [3^k + 3^{k-1}, 2\times 3^k]$ then there exist binary- and ternary-rooted optimal trees, (2) if $n \in [3^k, 3^k + 3^{k-1} - 1] \cup [2 \times 3^k + 1, 3^{k+1} - 1]$ then the only optimal tree is ternary-rooted, (3) $\kappa'(d)$ is weakly increasing in d and, for any integer $k \geq 1$, marginal costs are constant between 3^k and 2×3^k and between 2×3^k and 3^{k+1}. Proofs of (1) and (2) are in Göbel and Hoede (1979), and of (3) in Knuth (1998b).

Corollary of (3): if the successors v_1 and v_2 of the root r of an optimal tree define

subtrees with n_1 and n_2 leaves respectively then there does not exist an integer $k > 0$ such that either (i) $n_1 < 2 \times 3^k$ and $n_2 > 2 \times 3^k$, or (ii) $n_1 < 3^k$ and $n_2 > 3^k$. For case (i), if there were such a k then (3) would imply $\kappa'(n_2 - 1) > \kappa'(n_1)$ and hence a cost reduction could be achieved by letting the two successors of r define trees with $n_1 + 1$ and $n_2 - 1$ leaves. The proof for case (ii) is similar.

Let T be an optimal tree. We first show that T cannot have any nodes with $l > 4$ successors. Since $3^{\lceil \log_3 l \rceil} \geq l$ for any $l \geq 1$, there is a tree with l leaves such that each path consists of $\lfloor \log_3 l \rfloor$ ternary nodes and then either a leaf or a binary or ternary node followed by a leaf. If $l > 6$ then $3 \lceil \log_3 l \rceil < l$ (see the proof of Theorem 2) and hence any path of this tree has a cost less than l. We can exclude a node v with 6 or 5 successors due to the following replacements of v and its successors having weakly smaller cost on every path and strictly smaller cost on some path. For $l = 6$, the replacement is a ternary node followed by three binary nodes and their successors, and for $l = 5$ it is a ternary node followed by a singleton successor and two binary nodes and their successors.

If some v in T has 4 successors then replacement of v and its successors by a binary node followed by two binary nodes and their successors does not change the cost of any path and at least weakly increases the number of nonternary nodes on any path. We may therefore assume that T consists solely of binary and ternary nodes.

For any node v in T, let n_v denote the number of leaves in the subtree of T with root v.

Consider a node v in T that is binary and such that no other node on the sequence between the root r and v is also binary. Let $k = \lfloor \log_3 n_v \rfloor$ be given by the representation $n_v = 3^k + s$. Since v is binary, (2) implies that $n_v \geq 3^k + 3^{k-1} = 4 \times 3^{k-1}$. If u is one of the successors of v and $n_u < 2 \times 3^{k-1}$ then, letting u' be the other successor, we have $n_u + n_{u'} = n_v \geq 4 \times 3^{k-1}$ and therefore $n_{u'} > 2 \times 3^{k-1}$, in violation of the Corollary. Similarly, since $n_v \leq 2 \times 3^k$ by (2), $n_u > 3^k$ for one successor u would imply $n_{u'} < 3^k$ for the other successor u', again violating the Corollary. So $n_u \in [2 \times 3^{k-1}, 3^k]$ for any successor u of v.

Next consider the nonleaf and nonroot nodes w in the subtree T' of T with root v. In each case, $k_w = \lfloor \log_3 n_w \rfloor$ will indicate the value given by the representation $n_w = 3^{k_w} + s$.

A. Suppose $n_w = 3^{k_w}$. Then by (2) w is ternary and, due to the Corollary, there cannot be a successor u node of w such that $n_u < 3^{k_w - 1}$ (resp. $n_u > 3^{k_w - 1}$) since then there also would be a successor u' such that $n_{u'} > 3^{k_w - 1}$ (resp. $n_{u'} < 3^{k_w - 1}$). So, for each successor u of w, $n_u = 3^{k_w - 1}$. Hence all nodes on a path of the subtree with root w to a leaf of T' must be ternary.

B. Next suppose $n_w = 3^{k_w} + 3^{k_w} = 2 \times 3^{k_w}$. If w is ternary then, due to the Corollary, each successor u of w must have $n_u = 2 \times 3^{k_w - 1}$ while if w is binary then, again due to the Corollary, each successor u must have $n_u = 3^{k_w}$. Given paragraph A, we conclude that any path on the subtree with root w to a leaf of T' can pass through at most one binary node.

C. Finally when $n_w \in (2 \times 3^{k_w}, 3^{k_w + 1})$ then, due to (2), w must be ternary. Applying the Corollary twice, each successor u of w must have $n_u \in [2 \times 3^{k_w - 1}, 3^{k_w}]$.

To conclude, let $v, v_1, ..., v_m$ be a path on the subtree with root v to a leaf v_m of T' and let $1 \leq j \leq m$ be the greatest index such that n_{v_l} lies in $(2 \times 3^{k_{v_l}}, 3^{k_{v_l} + 1})$ for $1 \leq l < j$. Due to C, the nodes $v_1, ..., v_{j-1}$ are ternary. Due to A and B, at most one of the nodes

$v_j, ..., v_{m-1}$ can be binary. ■

Proof of Theorem 6. The proof of Theorem 3 shows that, for any integer $k > 3$, $k - 1 > \lceil \log_2 k \rceil$ and therefore $k\,(k-1) > k \lceil \log_2 k \rceil$. Hence replacing a node with k answers with an all-binary subtree of length $\lceil \log_2 k \rceil$ will strictly reduce $\kappa_E^*(T) = \sum_{l \in L(T)} \sum_{v \in p(l)} (k_v - 1)$ and weakly reduce $\kappa_M^*(T) = \max_{l \in L(T)} \sum_{v \in p(l)} (k_v - 1)$. Since $2^{\lceil \log_2 k \rceil} \geq k$, the replacement subtree can be pruned as necessary so that it has k leaves, which ensures that the replacement tree as a whole has n nodes. If a node v' has 3 children/answers then we can replace node v' and its answers, which we can view as a tree T', with a subtree T'' with 3 leaves that consists of a binary node followed by a binary node and a leaf. The subtree T'' reduces κ_E^* and does not increase κ_M^* since each leaf l of T'' has a cost $\sum_{v \in p_{T''}(l)} (k_v - 1)$ that is less than or equal to the cost $\sum_{v \in p_{T'}(l')} (k_v - 1) = 2$ of each leaf l' of T' and one leaf of T'' has cost 1 (where $p_{\widehat{T}}(\widehat{l})$ denotes the path from root to $\widehat{l}$ in tree $\widehat{T}$). Thus an expectation H-optimal tree must be all-binary and there is an all binary minimax H-optimal tree. Moreover, if T is expectation H-optimal then all paths of T must have lengths that differ by at most 1: if path p has length l and p' has length $l' > l + 1$ then moving a binary question from end of p' to the end of p would change κ_E^* by $2(l+1) - 2l' < 0$. Since any two trees are graph isomorphic if each is all-binary, has n leaves, and has nearly equal path lengths (and since expectation H-optimal trees exist), any all-binary tree T^* with n leaves and equal path lengths is expectation H-optimal. Since the maximum path length of T^* is $\lceil \log_2 n \rceil$ and any all-binary tree with n leaves must have a path with length of at least $\lceil \log_2 n \rceil$, T^* is also minimax H-optimal.

Now suppose T is minimax H-optimal and that $k_1^*, ..., k_{N^*}^*$ is a maximum-cost path. As in the proof the Lemma, $k_1^*, ..., k_{N^*}^*$ must solve $\min_{N \in \mathbb{Z}_{++}, (k_1, ..., k_N) \in \mathbb{Z}_{++}^N} \sum_{i=1}^N (k_i - 1)$ s.t. $k_1 \times ... \times k_N \geq n$. The proof of Theorem 3 shows that if some k_j^* does not equal 2 or 3 then replacing k_j^* with $\lceil \log_2 k_j^* \rceil$ components equal to 2 will reduce the objective function in this minimization problem. Specifically, $k_j^* - 1 > \lceil \log_2 k_j^* \rceil$ and therefore $\sum_{i=1}^{N^*} (k_i^* - 1) > \lceil \log_2 k_j^* \rceil + \sum_{i \neq j} (k_i^* - 1)$. And since $2^{\lceil \log_2 k_j^* \rceil} \geq k_j^*$, $2^{\lceil \log_2 k_j^* \rceil} \prod_{i \neq j} k_i^* \geq \prod_{i=1}^{N^*} k_i^*$.

Finally suppose there are $m > 2$ indices i such that $k_i^* = 3$. Since the proof of Theorem 3 shows that $2 \lceil \log_3 n \rceil > \lceil \log_2 n \rceil$ for $n > 14$, we conclude that $2 \lceil \log_3 3^m \rceil > \lceil \log_2 3^m \rceil$. Since $2^{\lceil \log_2 3^m \rceil} \geq 3^m$, we have $2^{\lceil \log_2 3^m \rceil} \prod_{i: k_i \neq 3} k_i^* \geq \prod_{i=1}^{N^*} k_i^*$. Therefore $(k_1^*, ..., k_{N^*}^*)$ could not be a solution. ■

References

[1] Aigner M, 2007, *Discrete Mathematics*. American Mathematical Society: Providence RI.

[2] Bolton P and Dewatripont M, 1994, 'The firm as a communication network,' *Quarterly Journal of Economics* 109: 809-839.

[3] Conan Doyle A, 1890, 'The sign of the four,' *Lippincott's Monthly Magazine* 45, February, 145-223.

[4] Cowan N, 2000, 'The magical number 4 in short-term memory: a reconsideration of mental storage capacity,' *Behavioral and Brain Sciences* 24: 87-185.

[5] Crémer J, Garicano L, and Prat A, 2007, 'Language and the theory of the firm,' *Quarterly Journal of Economics* 122: 373-407.

[6] Dilmé F, 2017, 'Optimal languages,' mimeo, University of Bonn.

[7] Engineering Research Associates, Inc., 1950, *High-Speed Computing Devices.* McGraw-Hill: New York.

[8] Frieder G, 1972, 'Ternary computers, part 1: motivation and part 2: emulation' *Proceedings of the 5th Workshop on Microprogramming*, 83-89.

[9] Gigerenzer G, Todd P, and the ABC Research Group (1999) *Simple Heuristics That Make Us Smart.* Oxford University Press, New York.

[10] Göbel C and Hoede F, 1979, 'On an optimality property of ternary trees,' *Information and Control* 42: 10-26.

[11] Hallworth R and Heath F, 1961, 'Semiconductor circuits for ternary logic,' *The Institution of Electrical Engineers* Monograph No. 482 E.

[12] Hayes B, 2001, 'Third base,' *American Scientist* 89: 490-494.

[13] Huffman D, 1952, 'A method for the construction of minimum-redundancy codes,' *Proceedings of the Institute of Radio Engineers* 40: 1098-1101.

[14] Hurst S, 1984, 'Multiple-valued logic its status and its future' *IEEE Transactions on Computers* 33: 1160-1179.

[15] Kahneman D, Treisman A, and Gibbs B, 1992, 'The reviewing of object files: object specific integration of information' *Cognitive Psychology* 24: 175-219.

[16] Klimenko S, 1999, 'Computer science in Russia: a personal view' *IEEE Annals of the History of Computing* 29: 16-30.

[17] Knuth D, 1998a, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms.* Addison-Wesley: Reading MA.

[18] Knuth D, 1998b, *The Art of Computer Programming, Vol. 3: Sorting and Searching.* Addison-Wesley: Reading MA.

[19] Maćkowiak B, Matějka F, and Wiederholt M, 2018, 'Rational inattention: a review' *CEPR Discussion Papers* No. 15408.

[20] Miller G, 1956, 'The magical number seven, plus or minus two: some limits on our capacity for processing information' *Psychological Review* 63: 81-97.

[21] Parhami B and McKeown M, 2013, 'Arithmetic with binary-encoded balanced ternary numbers' *Proceedings of the 47th Asilomar Conference on Signals, Systems, and Computers*, 1130-1133.

[22] Radner R, 1993, 'The organization of decentralized information processing' *Econometrica* 61: 1109-1146.

[23] Simon H, 1974, 'How big is a chunk?' *Science* 183: 482-488.

[24] Sims C, 2003. 'Implications of rational inattention' *Journal of Monetary Economics* 50: 665-690.

[25] Sobel, J., 2015, 'Broad terms and organizational codes,' Technical report, UCSD.

[26] Treisman A, 2006, 'Object tokens, binding and visual memory,' in H. Zimmer, A. Mecklinger, and U. Lindenberger (eds.) *Handbook of Binding and Memory: Perspectives from Cognitive Neuroscience*. Oxford University Press, New York.