In Defense of the Unprovability of the Church-Turing Thesis

SELMER BRINGSJORD[1]*, NAVEEN SUNDAR G.

Department of Computer Science, Department of Cognitive Science,
Rensselaer AI & Reasoning Laboratory,
Rensselaer Polytechnic Institute (RPI), Troy, NY 12180, USA

One of us has previously argued that the Church-Turing Thesis (CTT), contra Elliot Mendelson, is not provable, and is — in light of the mind's ability to effortlessly hypercompute — moreover false. But a new, more serious challenge has appeared on the scene: an attempt by Peter Smith to prove CTT. His reasoning is an ingenious "squeezing argument" that makes crucial use of Kolmogorov-Uspenskii (KU) machines. We analyze Smith's case, and in light of three objections find it wanting. We end by briefly pointing to our next steps in a thorough evaluation of all published attempts to prove CTT.

Key words: Effective computation; Church-Turing thesis; Kolmogorov-Uspenskii machines; Hypercomputation; Squeezing argument

1 INTRODUCTION

One of us has previously argued that the Church-Turing Thesis (CTT), contra Elliot Mendelson, is not provable, and is — light of the mind's capacity for effortless hypercomputation — moreover false (e.g., [13]). But a new, more serious challenge has appeared on the scene: an attempt by Smith [28] to prove CTT. His case is a clever "squeezing argument" that makes crucial use of Kolmogorov-Uspenskii (KU) machines.

The plan for the present paper is as follows. After covering some necessary preliminaries regarding the nature of CTT, and taking note of the fact that this thesis is "intrinsically cognitive" (§2), we: sketch out, for context, an open-minded position on CTT and related matters (§3); explain the formal structure of squeezing arguments (§4); after a review of KU-machines, formalize Smith's case (§5); give our objections to certain assumptions in Smith's argument (§6); support these objections with some evidence from general but limited-agent problem solving (§7); and explain why Smith's argument is inconclusive (§8). We end with some brief, concluding remarks, some of which point toward near-future work that will build on the present paper (§9).

2 PRELIMINARIES

At the heart of CTT is the notion of an *algorithm*, characterized traditionally as a finite and completely specified step-by-step and initial-step-to-concluding-step procedure for solving an entire class of problems. An algorithm is prescribed in advance and does not depend upon any physical or random factors. While different words are used in different textbooks, the previous two sentences constitute an adequate distillative description of the concept in question. A function $f : A \to B$ is then called *effectively computable* iff there exists an algorithm that an idealized computing agent can follow in order to compute the value $f(a)$ for any given $a \in A$.[†] It is crucial to note the central

* email: `selmer@rpi.edu`

[†] We use the generic term 'computing agent,' or sometimes — following Smith — simply 'agent'. Turing [29] spoke of "computists" and Post [20] of "workers," humans whose sole job was to slavishly follow explicit, excruciatingly simple instructions.

role of the "idealized computing agent" in this characterization of effective computation. The centrality of such a creature makes effective computation an intrinsically *cognitive* concept — and Smith, given what he says in his case for CTT, agrees (e.g. see note ¶).

Without loss of generality, we can restrict attention to so-called *number-theoretic functions*, i.e., functions that take N to N (where N is the set of natural numbers). Briefly, the justification for this restriction is a technique known as *arithmetization*. Using ideas made popular by Gödel, one can devise encoding and decoding algorithms that will represent any finite mathematical structure (e.g., a graph, a context-free grammar, a formula of second-order logic, a Java program, etc.) by a unique natural number. By using such a scheme, a function from, say, Java programs to graphs, can be faithfully (and *effectively*) represented by a function from N to N. Similar techniques can be used to represent a function of multiple arguments by a single-argument function.

The notion of an effectively computable function is at least seemingly informal, since it is based on the somewhat vague concept of an algorithm, and on the not-exactly-transparent concept of a cognitive agent. CTT also involves a more formal notion, that of a *Turing-computable* function. A (total) function $f : N \rightarrow N$ is Turing-computable iff there exists a Turing machine which, starting with n on its tape (perhaps represented by n |s), leaves $f(n)$ on its tape after processing, for any $n \in N$. (The details of the processing are harmlessly left aside for now; see, e.g., Lewis & Papadimitriou [18] for a thorough development.) Given this definition, CTT amounts to:

CTT A function $f : N \rightarrow N$ is effectively computable if and only if it is Turing-computable.

3 A NOTE ON OPEN-MINDEDNESS

For the present paper, we do not assume that CTT is true, nor do we assume its falsity. In addition, while we do assume that, in the abstract, there certainly are problem-solvers with powers far exceeding those with which human persons are blessed, we do not assume that human problem-solvers are incapable of information processing above the so-called *Turing Limit*.[⋆] Finally, we assume that our readers are similarly open-minded. If someone *knows* CTT, then of course it follows (by the unexceptionable dictum that knowing p entails the truth of p) that CTT is true, and every true proposition is the conclusion of a formally valid argument having true premises — so the upshot of a presupposition that CTT is known (by e.g. Smith) would be very close to the precise point at issue, viz., whether CTT *is* provable.

Open-mindedness as we have described it in the present section will turn out to be important when assessing Smith's purported proof of CTT.

4 THE FORMAL STRUCTURE OF "SQUEEZING" ARGUMENTS

The structure of a "squeezing" argument seems to have been originated by Kriesel [17], when he argues that the intuitive property of *being logically valid in virtue of form alone* can be identified with the formal property of *being proof-theoretically valid in a standard proof calculus c for first-order logic*.[†] The argument for this identification, essentially following Smith, can be set out as follows.

We allow quantification over arguments $\alpha_1, \alpha_2, \ldots$, and label the first of the two key properties in the previous paragraph as V_{int}, and the second as V_{formal}.[‡] We shall assume that each α_i is composed in part of a set of premises Φ_i and a conclusion χ_i. Using customary notation, we note that

$$V_{formal}(\alpha_i) \text{ iff } \Phi_i \vdash \chi_i$$

[⋆] Discussion of this limit can e.g. be found in Siegelmann [26]. By the way, our guess is that Smith is a computationalist: i.e., he holds that human persons are biological computers no more powerful than Turing machines. If computationalism is false — and as a matter of fact one of us has offered more than twenty deductive arguments designed to demonstrate the falsity of computationalism (e.g., starting with [8]) — then it seems reasonable to hold that it would seem to human persons that some of their hypercomputational problem-solving moves are automatic and instinctive, and hence in the space of the kind of "algorithmic" problem-solving that Smith is focused upon in his case for CTT. Relevant here is Bringsjord's claim that judgments about whether or not stories are truly interesting are effective/algorithmic and yet hypercomputational [13].

[†] The proof calculus c can e.g. be a natural-deduction one, as in the calculus $\mathcal{F}$ set out in (Barwise & Etchemendy [4]), or a standard resolution one e.g. defined in (Russell & Norvig [23]). Sometimes in introductory formal logic the first property here is said to be possessed by arguments when they are such that, if their premises are true, their conclusion *must* be as well.

[‡] It is unrealistic to assume that there are only a countably infinite number of arguments, but to ease exposition, and without loss of generality on the issue at hand, we make that assumption.

holds by definition, and that the following conditional, which parallels the "easy" direction of CTT, seems quite undeniable.

1. If $V_{formal}(\alpha_i)$ then $V_{int}(\alpha_i)$.

Next, where $\mathcal{I}$ is an interpretation (in the standard model-theoretic sense associated with first-order logic; standard coverage is e.g. provided in Ebbinghaus et al. [15]), we can say that an argument α_i has property M provided that whenever $\mathcal{I} \models \Phi_i$ it's also true that $\mathcal{I} \models \chi_i$. When the interpretation in question has as its domain the natural numbers, we write $\mathcal{I}^{\mathbf{N}}$, and property $M^{\mathbf{N}}$ has the obvious meaning. In this context, Kreisel recommends the conditional

2. If $V_{int}(\alpha_i)$ then $M^{\mathbf{N}}(\alpha_i)$.

on the strength of its contrapositive ($2'$.).[¶] The basic idea is that it's supposed to be evident that if some argument has a countermodel in the natural numbers, then that argument cannot be valid in virtue of its form. But next, it's a well-known theorem that

3. If $M^{\mathbf{N}}(\alpha_i)$ then $V_{formal}(\alpha_i)$.

It follows from 1., 2., and 3. (indeed, the proof is trivial in any standard proof calculus for first-order logic) that

4. $V_{int}(\alpha_i)$ iff $V_{formal}(\alpha_i)$,

and hence Kreisel takes himself to have established that *valid-in-virtue-of-form* is a property that can be identified with formal, proof-theoretic validity.[⋆]

Now let's generalize.

We allow quantification over not just arguments, but any relevant class of objects $x_1, x_2, \ldots$; and we speak of three general relation symbols X_{int}, X_{formal}, and X'_{formal}. Then a squeezing argument for

$$X_{int}(x_i) \text{ iff } X_{formal}(x_i)$$

consists in establishing the following squeeze on the intuitive concept in question:

$$X_{formal}(x_i) \rightarrow X_{int}(x_i) \rightarrow X'_{formal}(x_i) \rightarrow X_{formal}(x_i).$$

In this chain, the conditional $X'_{formal}(x_i) \rightarrow X_{formal}(x_i)$ is an outright "certified" theorem.[†] In addition, the first conditional in the chain is generally supposed to be either provable or perhaps self-evident. Finally, often the conditional $X_{int}(x_i) \rightarrow X'_{formal}(x_i)$ is derived directly from the contrapositive of this conditional.

5 SETTING OUT SMITH'S SQUEEZING ARGUMENT

In this section, we formally set out Smith's squeezing argument for CTT. His argument makes use of a specific scheme of computing. Hence we make a brief digression to expound this scheme, and to also introduce the theme of satisfiability in a logical system, which, as will be soon seen, recurs after its introduction in the present section.[‡]

[¶] I.e.,

2′. If $\neg M^{\mathbf{N}}(\alpha_i)$ then $\neg V_{int}(\alpha_i)$..

[⋆] We are not persuaded, but space does not permit a presentation of our objections. We will say only that Kreisel's argument seems to be overthrown by the fact that there are intuitively valid arguments (indeed, outright proofs) which cannot be represented in first-order logic. Examples might include arguments/proofs with highly expressive formulas (modal operators, e.g.), and arguments with visual information. To see examples of the former type, see [1]; and to see examples of the latter type see [3].

[†] Our requirements for some purported proof's being *certified* are explained in [2].

[‡] The sense of 'system' here used can be the limited one at the heart of Lindström's Theorems [15]. But we operate with an enlarged sense of 'logical system' in keeping with a cognitive perspective; that sense is characterized in [11].

5.1 KU Machines

In section 2 we reviewed the usual formal preliminaries (Turing machines, etc.). But Smith's argument makes crucial use of a type of computing machine he believes accords much better with the notion of a limited cognitive agent computing some function via some algorithm: viz., *Kolmogorov-Uspenskii machines*, or just *KU machines* for short.[¶] Accordingly, we give now an encapsulated account of KU machines, as they are defined by Smith.

KU machines were introduced by Kolmogorov & Uspenskii [16] to capture a general definition of algorithms. KU machines operate on a set of states S. The states are directed graphs with color-coded (or typed) edges. The nodes act as data cells and have symbols from a finite alphabet written in them. The nodes together with the edges constitute the *dataspace* of the machine. The edge colors are drawn from a fixed finite set. No two edges adjacent to a node can have the same color. The last two conditions limit the number of neighbors for any node. There is a unique *focal node*, f, at any stage in the computation, and revolving around the focal node is the *active patch*. The active patch consists of all the cells which are within a fixed distance, n (called the *attention span*) from the focal node. At any step in the computation, the next step depends solely on the active patch (and relevant instructions in the program). A KU machine program consists of a function γ with a finite domain. This is given in the form of a finite set of pairs

$$\{(P_1, \gamma(P_1)), (P_2, \gamma(P_2)), \ldots, (P_k, \gamma(P_k))\}.$$

The algorithm proceeds by replacing the active patch with $\gamma(P)$ if the active patch is isomorphic to P. Associated with each pair $(P_i, \gamma(P_i))$ is a mapping ϕ_i between the nodes in the boundary of the active patch P_i to certain nodes in $\gamma(P_i)$. The mapping ϕ_i ensures that the new active patch aligns with the rest of the dataspace. Initial states, I, and final states, F, are characterized by certain special symbols in the focal node f. The initial states contain an encoding of the input, and the final states contain an encoding of the output of the computation.

Smith accurately boils KU machines down to an elegant and laconic list of ten conditions; we do not go through all ten here, but discuss some of them below.

Example: Validity of an Argument in Propositional Calculus

We give a brief example to help explain KU machines. The example, which also serves to introduce the theme of satisfiability (which is, as we have said, a theme in the present paper), runs as follows.

Consider an argument α in the propositional calculus. The argument α consists of premises $\phi_1, \phi_2, \ldots, \phi_n$ and a conclusion ψ, and it is represented using traditional notation as $\{\phi_1, \phi_2, \ldots, \phi_n\} \vdash \psi$. The intuitive notion of validity $V_{int}(\alpha)$ holds iff the conclusion is true whenever the premises are true. The argument α is considered formally valid in the propositional calculus, $V_{formal}(\alpha)$, iff $\{\phi_1, \phi_2, \ldots, \phi_n, \neg\psi\}$ is not satisfiable.

There is a familiar algorithm, the *truth-tree* algorithm $\mathcal{T}$ [5], which can be used to test whether a given set of formulas in the propositional calculus is satisfiable. If we find that the set of formulas, $\Gamma = \{\phi_i | i = 1 \ldots n\} \cup \{\neg\psi\}$, comprising the premises and the negated conclusion, is not satisfiable, we can infer that the argument in question is valid, i.e., $\alpha = \{\phi_1, \phi_2, \ldots, \phi_n\} \vdash \psi$. The algorithm $\mathcal{T}$ works by constructing a *truth-tree* τ for the set of formulas Γ. The truth tree τ is a *rooted tree* and is constructed using the formulas in Γ as follows:

1. Place the root r.
2. The list of formulas in Γ is converted to a list of formulas Γ' such that Γ is not satisfiable iff Γ' is not satisfiable, and Γ' consists of formulas which are either disjuncts or conjuncts of literals. There is a straightforward algorithm to accomplish this.
3. For each formula in Γ':

[¶] Smith quite explicitly praises KU-machines over Turing machines because of the cognitive plausibility of the former. E.g., we read

> [W]hen we do real-life computations by hand, we often insert temporary pages as and where we need them and equally often throw away temporary working [sic] done earlier. . . . The good news is that this and other worries [about the Turing machine format] can be quieted by appeal to the very general condition for algorithmic computation given in 1958 by A. N. Kolmogorov and V. A. Uspenskii . . . [28, p. 333]

A note on notation: Following the tradition of using the string 'Turing machines' instead of 'Turing-machines,' we use 'KU machines.' The string 'KU-machine' then becomes an adjective. E.g., we could write something like 'c is a KU-machine cell.'

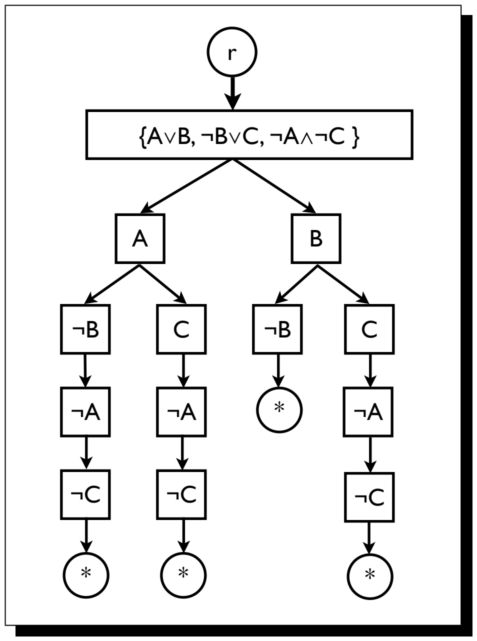
(a) If the formula is a disjunct of literals, for each leaf l in the unfinished tree place the literals as new leaves in a *parallel* fashion and connect each literal to the old leaf l.

(b) If the formula is a conjunct of literals, for each leaf l in the unfinished tree place the literals as new leaves in a *serial* fashion and connect the topmost literal to the old leaf l.

Figure 1 shows the tree for the argument $\{A \vee B, \neg B \vee C\} \vdash A \vee C$ after $\mathcal{T}$ has processed it (by adding flowers to the tree).

The set of formulas is unsatisfiable iff there is no path from the root of the tree to any of the leaves without an atom and its negation appearing in it. For each such path from the root to a leaf where an atom and its negation appear, $\mathcal{T}$ adds a extra leaf node containing a *flower* "*". If all the paths contain at least one such conflicting pair, the set of formulas is unsatisfiable. From this we can conclude that the argument α is indeed valid.

FIGURE 1
Result of the Algorithm $\mathcal{T}$

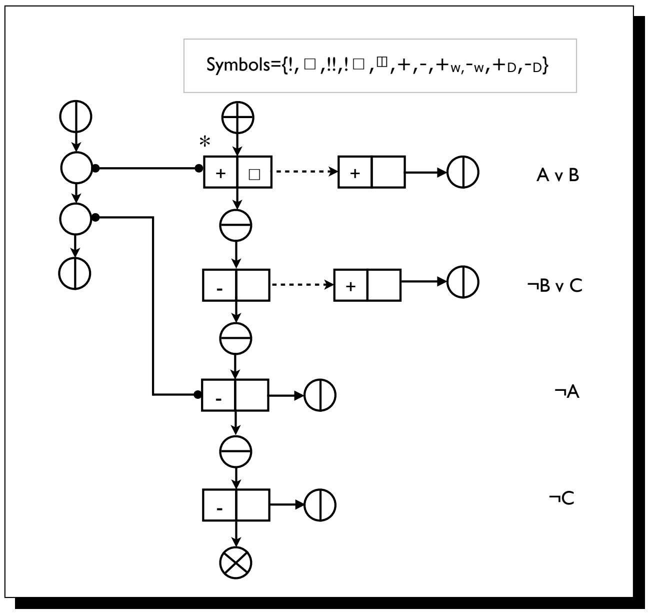


The decision problem solved by the truth-tree algorithm is a Turing-computable problem; hence there exists a KU machine such that when given the encoding of an argument, $encode(\alpha) \in I$, halts with the state $encode(0) \in F$ when the argument is valid and $encode(1) \in F$ when the argument is not valid. Here $encode$ is an appropriate encoding function (recall that we referred to Gödel numbering in §2). Since in any given argument there are only a finite number of atoms and a finite number of sentences, we can trivially encode an argument as a KU machine's initial state.

One possible KU-algorithm for this problem would construct the truth-tree starting out from the root r and proceed in depth-first fashion to examine all possible assignments. This is similar to fixing the truth-functional assignment for one literal and then trying out the other literals till either a conflict occurs or a satisfying assignment is found. If some assignment satisfies the set of formulas, the algorithm will then halt with the state $encode(0)$; else it will halt with the state $encode(1)$. This algorithm accords well with our intuition of how a human "computist" might solve such a problem.

Figure 2 represents an intermediate state of the machine for the argument in Figure 1 as input. Here the set of clauses is assumed to be in conjunctive normal form. Each clause is represented in a single horizontal line. The labels at the left denote the clause being represented. Each rectangular box denotes a literal. The first compartment in the box holds the sign of the literal and the second compartment holds temporary data. The two compartments can be

FIGURE 2
A KU-machine in Action



represented as two different cells joined by a single arrow. Each atom appearing in the argument is represented by a unidirectional chain of circles. The number of circles in the chain for an atom is equal to the number of literals the atom appears in. Each literal then connects to exactly one circle in the corresponding atom's chain of circles (in Figure 2 only the links for literals A and $\neg A$ are shown). The circles with bisecting lines and grids in them serve as convenient "end-markers." The asterisk appears next to the current focal node. The important point to note is that there exists a KU machine which solves this problem. The exact formulation of such a machine is left as an exercise for the reader.[⋆]

5.2 The Argument Itself

Smith's argument directly matches the structure of a squeezing argument as we presented it in section 4. With obvious symbolization, this means that the squeeze is accomplished when the following chain[†] is established:

$$Tcomputable(f) \rightarrow Effcomputable(f) \rightarrow KUcomputable(f) \rightarrow Tcomputable(f).$$

And as you will be able to infer, the chain is in turn established when the relevant trio of conditionals are established, to wit:

1⋆. If $Tcomputable(f)$ then $Effcomputable(f)$

2⋆. If $Effcomputable(f)$ then $KUcomputable(f)$

3⋆. If $KUcomputable(f)$ then $Tcomputable(f)$

Obviously, $\{1^\star., 2^\star., 3^\star.\} \vdash$ CTT.[‡] Conditional 3⋆. is provable. Conditional 1⋆. seems to us to be provable as well. (Remember that 1⋆. is a counterpart to the "easy" half of CTT, and Bringsjord has long ago conceded to Mendelson that

[⋆] See http://kryten.mm.rpi.edu/CTPROV/KU-TruthtreeAlgorithm.pdf for one such formulation.

[†] Smith prefers to start the chain with reference to μ-recursiveness rather than Turing-computability, but this needlessly complicates things. Besides, Smith immediately appeals to Turing machines in order to justify the first conditional: see section 35.5, p. 332 in [28].

[‡] The second of these conditionals, in keeping with our analysis of the counterpart conditionals in the above-visited squeezing argument from Kreisel, might be said to be derived from its contrapositive, but Smith, as we shall soon see when looking at what he says about 2⋆., makes no such move.

the easy half is indeed provable.) It thus follows that if Smith's argument fails, the source of the trouble must be $2^{\star}$. In light of this situation, does Smith himself defend $2^{\star}$.? Yes, he does. His defense is based, first, on the observation that if $2^{\star}$. is to be overthrown, that must happen because its antecedent holds, while its consequent doesn't. The second and final part of the defense is to argue that such a counter-example would need to be a case wherein an agent processes an algorithm effectively, but where there is some violation of the KU-machine format — and to then show that no such case is possible.

In support of this exegesis, we read:

> The KU specification involves a conjunction of requirements (finite alphabet, logically navigable workspace, etc.). So for a proposed algorithmic procedure to fail to be covered, it must falsify one of the conjuncts. But how? By having and using an infinite number of primitive symbols? Then it isn't usable by a limited computing agent like us (and we are trying to characterize the idea of an algorithmic procedure of the general type that agents like us could at least in principle deploy). By making use of a different sort of dataspace? But the KU specification only requires that the space has *some* structure which enables the data to be locally navigable by a limited agent. By not keeping the size of the active patch of dataspace bounded? But algorithms are supposed to proceed by the repetition of "small" operations which are readily surveyable by limited agents. By not keeping the jumps from one active patch of dataspace to the next active patch limited? But again, a limited agent couldn't then always jump to the next patch 'in one go' and still know where he was going. By the program that governs the updating of the dataspace having a different form? But KU algorithms are entirely freeform; there is no more generality to be had. [28, p. 337][¶]

These two points, then, should be entirely uncontroversial: If in fact there are cases where agents "like us" process algorithms in ways that violate the "KU conjunction," and these cases are found, Smith's case for CTT is refuted. And if *for all we know* such cases are theoretical possibilities, Smith's case for CTT is inconclusive. We proceed to give three objections, each of which, alone, implies the latter; and which, combined, seem to us to undeniably reveal Smith's case to be a very weak one.[⋆]

6 THREE OBJECTIONS: SMITH'S ARGUMENT IS INCONCLUSIVE

6.1 Objection #1: Artificially Legislated Boundedness

As we have seen, if even one of the ten conjuncts i.–x. which together compose specification of the KU framework is potentially violated by what limited agents like us do in the process of algorithmic problem-solving, Smith's purported proof of CTT is inconclusive. The third and sixth conjuncts/conditions are explicitly stated by Smith as follows.

> **iii.** At every stage in the computation of a particular algorithm, a patch of the dataspace of at most some fixed bounded size [$j \in \mathbf{N}$] is 'active.'
>
> **vi.** There is a fixed bound [$k \in \mathbf{N}$] on how far along the current network of cells the focal vertex of the active patch shifts from one step of the algorithm to the next. [28, p. 336]

Hence, in Smith's argument, strict bounds are placed on the capacity of the cognitive agent following the effective procedure. But why should it be acceptable to legislate that the size of the active patch remain constant at size j through some problem-solving run? Smith has an answer: "Because the maximum attention span of a limited cognitive agents stays fixed as he runs the algorithm (he doesn't get smarter!)." [28, p. 335]

Unfortunately, this doesn't make much cognitive sense, alas. An agent's attention span, in real life, fluctuates wildly through time, and not only that, but such fluctuations are not taken to signal that one's intelligence has actually changed. In fact, the usual situation is that in real life, limited problem-solvers like us may be compelled to premeditatedly enlarge their attention span if progress isn't being made.

Conjunct vi. likewise seems quite artificial. Why should it be that the agent isn't free to move the focal vertex in unbounded fashion from some time t_n in the computation to the time t_{n+1}? As far as we can tell after much reading and study of Smith's case, the only support for this restriction is in the quote above; specifically, in the rather

[¶] The restriction of the alphabet set to be finite comes from Turing [29] and is based on physical considerations. The other restrictions on the agent can be traced to Kolmogorov & Uspenskii [16].

[⋆] While we have additional objections, due to space constraints we give only three here.

mysterious comment: “But again, a limited agent couldn’t then always jump to the next patch ‘in one go’ and still know where he was going.” [28, p. 339] From the standpoint of what is even remotely cognitively plausible for limited agents like us, this seems odd because we make arbitrarily large jumps from the configuration of our infinite (or at least potentially infinite) workspace at t_n, to the configuration at t_{n+1}. And, even if we didn’t in fact make such jumps, there would be no principled reason why, from the standpoint of open-mindedness regarding effective computation and problem-solving, such jumps *might* be made.

It turns out that at least the kernel of our objection to conjuncts iii. and vi. was anticipated by Black [6], something we recently learned after formulating our objection. In fact, Smith cites Black in this regard, and attempts to rebut Black’s criticism. Here is Black’s objection, formatted slightly by Smith:

> Given the extreme generality of [their] notion of locality, I think it is fair to say that we have here a rigorous proof of Church’s thesis *if we can assume that bounded attention span is built into the intuitive notion of effective computation*. However, this is a rather big ‘if.’ . . . [I]t is unclear why the idealization which allows a potentially infinite passive memory (the unlimited supply of Turing-machine tape) should not be accompanied by a corresponding idealization allowing unlimited expansion of the amount of information which can be actively used in a single step. [6, p. 256]; emphasis his

Black is here referring by Smith’s ‘their’ to Kolmogorov and Uspenskii — but of course Smith allows the point to be directed to his case for CTT. Given what we have said, Black’s concern can be read as a concern about conjunct iii.

What is Smith’s rebuttal? The long and short of it is that he modifies the KU scheme: he allows the growth of the active patch, from one moment t_i to its successor t_{i+1} for the agent, to move beyond some fixed bound k to a primitive-recursive function. As he writes:

> So in fact we could tweak the definition of a KU-algorithm to allow the size of the active patch to grow, as fast as you like so long [as] the growth is governed by some primitively recursive function . . . [28, p. 341]

In fact, Smith is willing to even *further* relax the restriction on the attention span of the agent through time from one moment to the next by allowing only a μ-recursive cap on it, and he defends this cap as well. We see this in the following:

> You might object: “What right do we have to build recursive bounds into the (re)denition of a KU-algorithm, rather than perhaps faster-growing effectively computable bounds? We can’t assume these are the same without assuming the Thesis which we are trying to prove!” But against this we can rerun our earlier thought with even more confidence. It surely is quite inconsistent with the intuitive idea of an idiot-proof algorithm which proceeds by small steps, accessible to limited agents, that what is processed at each step should not just get ever bigger, but get bigger even faster than Ackermann-fast! [27, 5]

The relaxed bound requires that the elementary steps of the agent be recursive. Though this restraint is widespread throughout literature, there has not been any compelling justification for the same. As Sieg [25] notes: dagger*

> But no one gave convincing and non-circular reasons for the proposed rigorous restrictions on the steps that are permitted in calculations. [25, 6]

Nonetheless, Smith confidently asserts that his tweaking “should be more than enough to quiet Black’s worries” ([28, p. 341]). Unfortunately, Smith doesn’t seem to have read Black carefully. Notice that Black is making a connection between “*infinite* passive memory” (emphasis ours) and the possibility of a “*corresponding* idealization allowing *unlimited* expansion” of the active patch. Thus, Black appears to entertain a jump in the active patch that exceeds even a μ-recursive limitation.

Now, we are not saying that we know, let alone that you, reader, know, that jumps (from time t to its immediate successor) in the size of active content during human problem-solving “runs” sometimes exceed the jumps produced by the Ackermann function. We *are* saying that *for all anyone knows*, such jumps happen, especially in light of the kind of “Eureka!” phenomena that routinely occur during successful human problem-solving (e.g., see the example jump discussed in section 7.1, and the Penrosean one pointed to in footnote †). The burden of proof is on Smith to

* Sieg attempts to justify the restraints on the elementary steps with an appeal to physical intuition. This argument is to us unsatisfactory, as it renders CTT a physical hypothesis, but in the interests of space we leave these issues aside for another day and another paper.

show that his restrictions on the active patch must be in place in order for the KU framework to accurately reflect human problem-solving. Absent this demonstration, his case is inconclusive.

6.2 Objection #2: The Cognitive Possibility of the "Effective Infinitary"

For those open-minded about the possibility of hypercomputing minds, it is interesting to explicitly consider frameworks in which the customary bounds that enforce the Turing Limit are dissolved. In the context of KU machines, it would specifically seem worth considering problem-solving frameworks in which cells contain formulas that in and of themselves apparently break these bounds. Smith, of course, is completely closed-minded with respect to such breakage, as confirmed by this text, which we visited above: "So for a proposed algorithmic procedure to fail to be covered [by KU-machines], it must falsify one of the conjuncts. But how? By having and using an infinite number of primitive symbols? Then it isn't usable by a limited computing agent like us . . ." Clearly, Smith is of the view that a problem-solving system allowing an infinite number of primitive symbols is not appropriate when the agents involved are human persons. But is this view justified? Is it consistent with what human persons do when they problem-solve? Might it not be that limited problem-solving agents make use of techniques that are at once effective and infinitary? We are inclined to answer the first two questions in the negative, and the third in the affirmative.

For example, how might KU machines be modified to reflect steadfast open-mindedness regarding more expressive underlying languages? Well, consider the simple infinitary logic $\mathcal{L}_{\omega_1\omega}$, in which countably infinite conjunctions are allowed. Hence, cells in this case would be permitted to contain countably infinite expressions such as

$$\bigwedge \phi$$

and the equivalent

$$\phi_1 \wedge \phi_2 \wedge \ldots .$$

As is well-known, $\mathcal{L}_{\omega_1\omega}$ occupies a somewhat special place among infinitary logics, because inference in this system can be considered quite mechanistic. So for example we might have a cell like

$$\boxed{\bigvee \psi}$$

and the cell

$$\boxed{\neg\psi_{23}}$$

in the active patch at some time during the computation, and at the next moment move to

$$\boxed{\psi_1 \vee \psi_2 \vee \ldots \psi_{22} \vee \psi_{24} \vee \ldots}$$

This would seem to be perfectly standard reasoning for limited agents like us, and hence appears to be a suitable target for what a KU-like framework should accommodate — and yet such reasoning is over infinitely long expressions.*

Comments made in the final paragraph of section 6.1 apply *mutatis mutandis* to what we have said regarding infinitary logic and human problem-solving. In short, to repeat with but a touch of new emphasis: KU-like machines "boosted" in direction of $\mathcal{L}_{\omega_1\omega}$ *appear* to offer a framework suggested by aspects of human problem-solving. We do not claim that *in fact* such boosting is at present known to be necessary. Once again, the burden is on Smith to show that such boosting is inappropriate.

6.3 Objection #3: Why Not Infinite but Rule-Governed Workspaces?

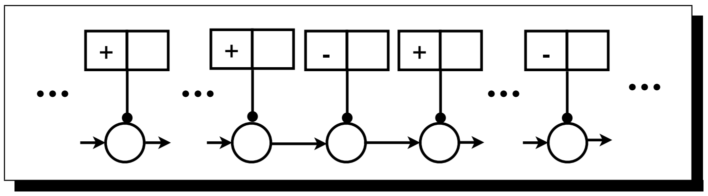
In section 5.1 we considered arguments composed of a single conclusion ψ and a finite number of premises

$$\{\phi_1, \phi_2, \ldots, \phi_n\}$$

* Predictably, some will point out that the *inscriptions* in cases such as this one are finite, since they fit into cells having a finite amount of space. However, infinitary logic is based on a fundamental distinction between such finitely long formulas as $\phi_1 \wedge \ldots \wedge \phi_m$ versus infinitely long formulas like $\bigwedge \phi$. This distinction is perfectly consistent with the fact that inscriptions in both cases are finite in size. For more on such issues, see the chapter on infinitary reasoning in [10], the ancestor of which is [9].

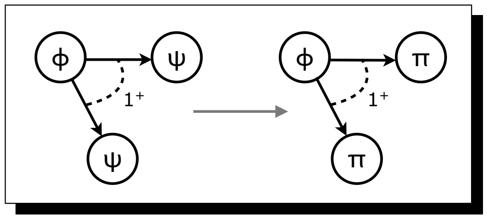
each of which are finitely long. The finiteness requirements then result in a finite number of atoms and a finite number of instantiations of each atom as a literal in the premises. When we remove the finiteness restrictions, we then operate on arguments with either a finite or infinite number of: premises, atoms and instantiations of each atom as literals in clauses. Consider an argument α which has an atom Q^{∞} instantiated an infinite number of times as a literal in the clauses. In Figure 2 we used a chain of nodes to represent an atom's literals, and each literal connects to exactly one node in its atom's chain. The nodes in the chain are used to keep track of the truth-functional assignment for that atom in the current search. The chain representing Q^{∞} then consists of an infinite number of nodes as shown in Figure 3. Even though the workspace for α is infinite in extent, we can have a simple mechanical procedure to find out whether α is valid. Since the infinitely long chain acts only as a bookkeeping gadget, one can extend the procedure for the finite workspace to lazily evaluate only needed portions of the chain.[†]

FIGURE 3
An Infinitely Long Chain in the Workspace



When we relax the conjuncts iii. and iv., we can make use of program *schemata* which are unbounded in form. Such schemata could correspond to an infinite number of $(P, \gamma(P))$ pairs when represented in the standard KU-machine formalism. The schema in Figure 4 gives one such example. In this schema, the cell containing ϕ is the focal node. The dataspace fragment on the right corresponds to P, and the one on the left corresponds to $\gamma(P)$. The schema is to be interpreted in the following manner: If the focal node is connected to one or more cells containing the formula ψ, replace the nodes containing ψ with nodes containing π.[⋆]

FIGURE 4
A Program Schema



7 EVIDENCE IN FAVOR OF LEAVING OUT BOUNDS

Limiting a general problem solver to a fixed cognitive capacity (a lá conjuncts iii., vi., and "tweaked" versions thereof) seems unnatural; and limiting a general problem solver to recursive growth seems even more unnatural and circular. Often in remarkably innovative leaps, humans learn and create new tools and representation schemes for problem solving. Our tools too improve in their capacity (e.g., the word size of microprocessors). So any computation scheme

[†] See http://kryten.mm.rpi.edu/CTPROV/KU-TruthtreeAlgorithm.pdf for a complete specification.

[⋆] This schemata violates the requirement that each arrow from a node be unique in color, but this does not prevent the schemata from being well-specified and does not prevent mechanization.

which purports to be the most general form of effective computation but factors out an expanding cognitive capacity is wholly unsatisfactory. In the present section, we give arguments to show why cognitive agents with unbounded capacity (armed therefore with environments beyond KU machines) might be useful and hence should be considered.

The fact is, even limited agents like ourselves, when following "standard procedure" in the process of seeking solutions to problems in the formal sciences, might well make shifts that exceed any fixed bound of size k. We believe that examples of such shifts are generally found when human problem-solvers jump from considering well-behaved finitary scenarios to infinitary ones. One of us has elsewhere discussed such examples,[†] and we now very briefly mention two here.

7.1 Case I: "Jumps" in Satisfiability Problems for First-Order Logic

Consider the *n-SAT* problem in first-order logic. A first-order formula ϕ valid in the domain of natural numbers $D = \{1...k\}$ for some $k \leq n$ is said to be n*-satisfiable*. This problem is Turing-computable. We can encode each ϕ using a unique natural number, and we can define the function $f_D\colon N \to \{0,1\}$ such that $f_D(m) = 1$ iff the formula represented by m is n-satisfiable. The function f_D is a Turing-computable function. For a given input formula, we test whether the formula holds for all the numbers in the finite domain D, and this can be accomplished in a finite number of steps using well-known algorithms (e.g. a readable presentation can be found in Boolos, Burgess & Jeffrey [7]). Moreover, this function can be computed by a KU machine with a bounded attention span (in light of conditional $3^\star$.).

But now consider the satisfiability problem, *SAT*, in general. (Not to be confused with the famous NP-complete problem of determining satisfiability for the propositional calculus.) Our agent receives a set Φ of first-order formulas and is charged with supplying an interpretation $\mathcal{I} \models \Phi$ if the set is satisfiable, and a simple "No" otherwise. This problem is Turing-unsolvable. However, a clever human problem-solver faced with SAT, when specifically faced with a set Γ satisfiable only by infinite interpretations, would try the first few small, finite elements of $\mathcal{P}(N)$ (power set of N), and when the formula fails to be modeled would then try to make an *unrestricted* jump to a larger but still finite $A \in \mathcal{P}(N)$. This process should go on for an unrestricted number of steps, and in each step the next trial set should not be restricted based on the current trial set. At a certain point, our human will suddenly begin to try to see if Γ can be satisfied by an interpretation having an infinite domain, starting with the domain of N. Our human problem-solver would seemingly make discrete but *unbounded* leaps. For an example of a system that in part works in this way, see [24].[⋆]

7.2 Case II: Generating Turing Machine Programs

Consider the problem of generating computer programs that compute arbitrarily given number-theoretic functions. The human problem solver — α_h — in this case (to use the common phrase, a *computer programmer*), has a finite (and suitably indexed) library $\mathcal{L}_h$ of programs $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$ which compute known functions. When a new, arbitrary function f (assuming that it's challenging to some degree) is supplied, α_h first checks whether there is a program $\mathcal{P}_f$ in $\mathcal{L}_h$ that computes f. If there is no such program, the programmer then generates a *new* program $\mathcal{P}_f$ and adds it to the library. Success in either case consists in the fact that (using obvious notation)

$$f(n) = \mathcal{P}_f[n].$$

Described in these admittedly broad, high-level terms, computer programming as problem-solving seems quite straightforward. Computational cognoscenti, however, know that nothing could be further from the truth. What *is* straightforward is how they know this. They know it because, for example, computer programmers, in the general case, must have the capacity to decide whether or not two computer programs compute the same function. If we move from talking of computer programs in general, to talking of Turing machines, then, formalized, where $H(n, k, u, v)$ holds if and only if Turing machine n halts in k steps with output v given input u, cognoscenti understand the severe

[†] There is for example the Penrosean example of problem solver suddenly grasping that adding together successive hexagonal numbers always results in cubes; see Penrose [19]. And there is the example of problem-solvers realizing that certain things hold in the limit; e.g., see Bringsjord & Arkoudas [12].

[⋆] We are not claiming that any particular human being can as a matter of empirical fact solve the satisfiability problem for first-order logic, but merely that it is more natural to think of human SAT problem-solvers operating in the fashion we describe.

challenge of computer programming because they see that computer programmers (in the general case) must ascertain whether or not this Π_2 formula holds (for suitable input):

$$\pi := \forall u \forall v[\exists k H(n,k,u,v) \leftrightarrow \exists k' H(m,k',u,v)]$$

Given this, what shall we say about the Smithian KU-based problem-solving framework? Consider an algorithmic agent α_a intended to model a human computer programmer. The agent α_a takes in as input the definition of a function f and is to generate a Turing machine m which computes f; that is, an m such that

$$f(n) = m[n].$$

The agent α_a can be assumed to have a collection of function-machine pairs:

$$\mathcal{L}_a = \{(f_1, m_1), (f_2, m_2), \ldots, (f_p, M_p)\} \text{ such that } f_i(n) = m_i[n]$$

And this agent can be assumed to have the power to decide whether input f matches some f_i in $\mathcal{L}_a$. But this part is a breeze. What about the rest — the part that would be reflective of the challenge expressed by π? We submit that this part of the process appears to involve leaps in innovation, and given that automatic programming (AP), the sub-field of AI and computer science devoted to engineering an α_a with the power of α_h, has been by any metric a dismal failure, there is no reason to believe, given open-mindedness about the issues before us (recall again section 3), that such leaps fit within the constraints of a KU machine as a framework for problem solving by limited agents like us.[†]

8 SMITH'S ARGUMENT IS INCONCLUSIVE

Recapping, Objection #1 explains why limiting cognitive agents to bounded capacity is unsatisfactory. As we noted above, though Black questions why "potentially unlimited" steps are not allowed, we only seek a smaller allowance: finite but large steps. Smith's arguments, contrary to what he claims, does not "quiet Black's worries."

Objection # 2 shows that even limited cognitive agents can apparently reason over infinitely long strings. We explained that a limited agent reasoning over infinitely long expressions violates some of the KU-defining conjunctions given by Smith.

Objection # 3 relates plausible ways in which even limited cognitive agents could mechanically use infinitely long constructs. Objection #3 was a response to Smith's continual assertion that limited agents are incapable of processing infinitary constructs in KU-style formats. The only reasoning offered by Smith to support this assertion is circular.

Each of the above three objections individually show that premise $2^\star$ in the key equation

$$\{1^\star., 2^\star., 3^\star.\} \vdash \text{ CTT}$$

is controversial, at best. Hence we have no reason to believe that this equation holds. Yet this equation is precisely what Smith needs to make his case. We thus conclude that Smith's case for CTT is inconclusive at best.

9 CONCLUSION AND FUTURE WORK

As pointed out earlier, one of us had previously refuted (by our lights, anyway) a pre-Smith attempt to prove CTT. The present paper is the first step in a program to overthrow Smith's attempt to prove CTT. We have in hand a number of additional objections to Smith's "proof," and are in the process of refining them. But in the future we also plan to consider other attempts to establish CTT (or a close relative); for example, one undertaken by Dershowitz & Gurevich [14], who derive CTT from a set of axioms they believe capture the essence of computability. Perhaps

[†] It would be well beyond the scope of the present paper to detail the miserable failure of AP when measured against the human case. While the aims of AP have fluctuated considerably over the decades, as has been pointed out by Rich & Waters [22] and others, the "black-box" version of AP, wherein the definition of a function f is provided as input, and the desired output is a program $\mathcal{P}_f$ that computes f, continues to be stalled, whether the approach is inductive or deductive.

unsurprisingly, we aren't persuaded by the case D&G present, but our refutation will need to be wait for another day. Ultimately, our research along the direction of CTT and formalisms intended to provide a cognitively plausible framework for effective problem-solving will, we believe, eventuate in modifications to these formalisms that include hypercomputational elements. This research is based not only on the foregoing analysis, and cited prior work within it, but on our conviction that KU machines, while certainly allowing some cognitively desirable features into a problem-solver's workspace, are in the end unwisely tethered to Turing machines, as is reflected in the provable $3^{\star}$. Constraining every Turing machine, as is well-known, is Rado's max-shift function S [21], which yields for a given n-state Turing machine the maximum number of shifts $S(n)$ it can make. From the open-minded, cognitive point of view, this seems like a straitjacket: When a problem-solver tackles some problem, it seems to us that he or she, over any session, is capable of an arbitrarily large number of moves.

REFERENCES

[1] K. Arkoudas and S. Bringsjord. (2008). Toward Formalizing Common-Sense Psychology: An Analysis of the False-Belief Task. In T.-B. Ho and Z.-H. Zhou, editors, *Proceedings of the Tenth Pacific Rim International Conference on Artificial Intelligence (PRICAI 2008)*, number 5351 in Lecture Notes in Artificial Intelligence (LNAI), pages 17–29. Springer-Verlag.

[2] Konstantine Arkoudas and Selmer Bringsjord. (2007). Computers, justification, and mathematical knowledge. *Minds and Machines*, 17(2):185–202.

[3] Konstantine Arkoudas and Selmer Bringsjord. (2009). Vivid: An AI framework for heterogeneous problem solving. *Artificial Intelligence*, 173(15):1367–1405. The url http://kryten.mm.rpi.edu/vivid/vivid.pdf provides a preprint of the penultimate draft only. If for some reason it is not working, please contact either author directly by email.

[4] J. Barwise and J. Etchemendy. (1999). *Language, Proof, and Logic*. Seven Bridges, New York, NY.

[5] Merrie Bergmann, James Moor, and Jack Nelson. (1997). *The Logic Book*. McGraw Hill, New York, NY.

[6] Robert Black. (2000). Proving church's thesis. *Philosophica Mathematica*, 8:244–258.

[7] G. S. Boolos, J. P. Burgess, and R. C. Jeffrey. (2003). *Computability and Logic (Fourth Edition)*. Cambridge University Press, Cambridge, UK.

[8] S. Bringsjord. (1992). *What Robots Can and Can't Be*. Kluwer, Dordrecht, The Netherlands.

[9] S. Bringsjord. (1997). An argument for the uncomputability of infinitary mathematical expertise. In P. Feltovich, K. Ford, and P. Hayes, editors, *Expertise in Context*, pages 475–497. AAAI Press, Menlo Park, CA.

[10] S. Bringsjord and M. Zenzen. (2003). *Superminds: People Harness Hypercomputation, and More*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

[11] Selmer Bringsjord. (2008). Declarative/logic-based cognitive modeling. In Ron Sun, editor, *The Handbook of Computational Psychology*, pages 127–169. Cambridge University Press, Cambridge, UK.

[12] Selmer Bringsjord and Konstantine Arkoudas. (2004). The modal argument for hypercomputing minds. *Theoretical Computer Science*, 317:167–190.

[13] Selmer Bringsjord and Konstantine Arkoudas. (2006). On the provability, veracity, and AI-relevance of the church-turing thesis. In A. Olszewski, J. Wolenski, and R. Janusz, editors, *Church's Thesis After 70 Years*, pages 66–118. Ontos Verlag, Frankfurt, Germany. This book is in the series *Mathematical Logic*, edited by W. Pohlers, T. Scanlon, E. Schimmerling, R. Schindler, and H. Schwichtenberg.

[14] Nachum Dershowitz and Yuri Gurevich. (2008). A natural axiomatization of computability and proof of church's thesis. *The Bulletin of Symbolic Logic*, 14(3):299–350.

[15] H. D. Ebbinghaus, J. Flum, and W. Thomas. (1994). *Mathematical Logic (second edition)*. Springer-Verlag, New York, NY.

[16] A.N. Kolmogorov and V.A. Uspenskii. (1958). On the definition of an algorithm. *Uspekhi Matematicheskikh Nauk*, 13(4):3–28.

[17] Georg Kreisel. (1967). Informal rigor and completeness proofs. In *Problems in the Philosophy of Mathematics*, pages 138–186. North-Holland, Amsterdam, The Netherlands.

[18] Harry R. Lewis and Christos H. Papadimitriou. (1997). *Elements of the Theory of Computation*. Prentice Hall.

[19] R. Penrose. (1994). *Shadows of the Mind*. Oxford, Oxford, UK.

[20] E. Post. (1944). Recursively enumerable sets of positive integers and their decision problems. *Bulletin of the American Mathematical Society*, 50:284–316.

[21] T. Rado. (1962). On non-computable functions. *Bell System Technical Journal*, 41(3):877–884.

[22] Charles Rich and Richard C. Waters. (August 1988). Automatic programming: Myths and prospects. *Computer*, 21(8):40–51.

[23] S. Russell and P. Norvig. (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ.

[24] Andrew Shilliday. (2009). *Elisa: A new system for AI-assisted logico-mathematical scientific discovery incorporating novel techniques in infinite model finding*. PhD thesis, Rensselaer Polytechnic Institute.

[25] W. Sieg. (2008). Church without Dogma-Axioms for computability. *New Computational Paradigms: Changing Conceptions of What is Computable*, pages 18–44.

[26] H. T. Siegelmann. (1999). *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhäuser, Boston, MA.

[27] P. Smith, (2009). Corrections to IGT.

[28] Peter Smith. (2007). *An Introduction to Gödel's Theorems*. Cambridge University Press, Cambridge, UK.

[29] A. M. Turing. (1936). On computable numbers with applications to the entscheidung-problem. *Proceedings of the London Mathematical Society*, 42:230–265.